Red Hat Linux - The Complete Reference, Second Edition

Richard Petersen

McGraw-Hill/Osborne
2600 Tenth Street
Berkeley, California 94710
U.S.A.

To arrange bulk purchase discounts for sales promotions, premiums, or fund-raisers, please contact McGraw-Hill/Osborne at the above address. For information on translations or book distributors outside the U.S.A., please see the International Contact Information page immediately following the index of this book.

Red Hat Linux: The Complete Reference, Second Edition

Copyright © 2002 by The McGraw-Hill Companies. All rights reserved. Printed in the United States of America. Except as permitted under the Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher, with the exception that the program listings may be entered, stored, and executed in a computer system, but they may not be reproduced for publication.

1234567890 CUS CUS 01987654321

Book p/n 0-07-219179-1, CD 1 p/n: 0-07-219180-5, and CD 2 p/n: 0-07-222386-3
parts of
ISBN 0-07-219178-3

Publisher
Brandon A. Nordin

Vice President & Associate Publisher
Scott Rogers

Acquisitions Editor
Francis Kelly

Project Editors
Mark Karmendy
LeeAnn Pickrell

Acquisitions Coordinator
Alexander Corona

Technical Editor
Dean Henrichsmeyer

Copy Editor
Dennis Weaver

Proofreaders
John Gildersleeve, Paul Tyler

Indexer
Jack Lewis

Computer Designers
Tara A. Davis, Lauren McCarthy

Illustrators
Lyssa Wald, Michael Mueller

Series Design
Peter F. Hancik

This book was composed with Corel VENTURA™ Publisher.

Information has been obtained by McGraw-Hill/Osborne from sources believed to be reliable. However, because of the possibility of human or mechanical error by our sources, McGraw-Hill/Osborne, or others, McGraw-Hill/Osborne does not guarantee the accuracy, adequacy, or completeness of any information and is not responsible for any errors or omissions or the results obtained from use of such information.

To my sisters-in-law,
Marylou and Valerie

About the Author

Richard Petersen holds an M.L.I.S. in library and information studies. He currently teaches Unix and C/C++ courses at the University of California, Berkeley.

Acknowledgments

I would like to thank all those at McGraw-Hill/Osborne who made this book a reality, particularly Francis Kelly, acquisition editor, for his continued encouragement and analysis as well as management of such a complex project; Dean Henrichsmeyer, the technical editor, whose analysis and suggestions proved very insightful and helpful; Alexander Corona, acquisitions coordinator, who provided needed resources and helpful advice; Dennis Weaver, copy editor, for his excellent job editing as well as insightful comments; and project editors Mark Karmendy and LeeAnn Pickrell who incorporated the large number of features found in this book as well as coordinating the intricate task of generating the final version. Thanks also to Scott Rogers and Jane Brownlow who initiated the project.

Special thanks to Linus Torvalds, the creator of Linux, and to those who continue to develop Linux as an open, professional, and effective operating system accessible to anyone. Thanks also to the academic community whose special dedication has developed Unix as a flexible and versatile operating system. I would also like to thank professors and students at the

University of California, Berkeley, for the experience and support in developing new and different ways of understanding operating system technologies.

I would also like to thank my parents, George and Cecelia, and my brothers, George, Robert, and Mark, for their support and encouragement with such a difficult project. Also Valerie and Marylou and my nieces and nephews, Aleina, Larisa, Justin, Christopher, and Dylan, for their support and deadline reminders.

About the CD-ROMs

The Linux distribution CD-ROM, Red Hat Linux 7.2, is included in the book, featuring the 2.4 kernel and Gnome 1.4. The Red Hat Linux distribution installs a professional-level and very stable Linux system with KDE and Gnome GUI interfaces, providing you with all the advantages of a Unix workstation on your PC, combined with the same ease of use and versatility found on GUI systems like Windows and Mac/OS. An extensive set of Internet servers are also included that are automatically installed along with flexible and easy-to-use system configuration tools. You can find recent information about Red Hat at **www.redhat.com**.

The CDs include both Gnome and the K Desktop Environment (KDE) GUI user interfaces, along with an extensive number of Gnome and KDE applications. Red Hat 7.2 installs both Gnome and KDE, as well as a comprehensive set of Linux software applications including the GNU software packages (graphics, communications, publishing, editing, programming, games), as well as development tools, and Internet servers (ftp, Web, mail, news, and DNS). It also installs a complete set of Internet clients such as mail, news, FTP, and Web browsers. There are clients for both the Gnome and KDE desktops, as well as for shell and window manger interfaces. With your installation, you are entitled to join the Red Hat Network, which provides you with automatic updates for your installed software and system. Periodically you can check for updates. The needed software updates will be detected, downloaded, and installed for you.

The CD-ROM and the Red Hat Web site (**www.redhat.com**) include extensive documentation including HOW-TO documents, tutorials in Web page format, and online manuals. Three very helpful Red Hat guides are the *Red Hat Installation Guide*, the *Red Hat Getting Started Guide*, the *Red Hat Linux Reference Guide*, and the *Red Hat Customization Guide.* All are in Web-page format and can be viewed with any Web browser on any system. The *Red Hat Installation Guide* provides a detailed walk-through of the installation procedure with graphics and helpful suggestions. It is best to check it before you install Red Hat. The *Red Hat Getting Started Guide* provides an overview of basic Red Hat operations such as working with Gnome and basic configuration. The *Red Hat Linux Reference Guide* covers administration and configuration tasks in detail. The *Red Hat Linux Customization Guide* covers specialize administration tasks, such as DNS, NFS, and Samba configuration, along with using encryption and the Red Hat Package manager.

For added functionality, you can also download free personal editions of the StarOffice office suite from **www.sun.com** and WordPerfect from **linux.corel.com**. Also, the Java Software Development Kit is available for free at **www.blackdown.org**. Databases are available from their respective Web sites, such as Oracle from **www.oracle.com**. Numerous applications, in the easy-to-install RPM package format, are available on the CD-ROM included with this book, and can also be downloaded from the Red Hat ftp site and its mirror sites,

ftp.redhat.com. You can both download and install these applications using either the Gnome or KDE file managers. Several popular Internet sites where you can easily obtain Linux applications are listed here.

Linux Applications	**Internet Site**
Red Hat FTP site	ftp.redhat.com (for manual updates or added software)
Linux Documentation Project	www.linuxdoc.org
Java Software Development Kit	www.blackdown.org
Window manger and desktop themes	www.themes.org
Gnome applications	www.gnome.org
KDE applications	apps.kde.com
Linux Application in RPM packages	www.rpmfind.net
New Linux applications	www.freshmeat.net
Linux opensource applications and development sites	www.sourceforge.net
Linux applications	www.linuxapps.com
WordPerfect	linux.corel.com
Linux World	www.linuxworld.com
Linux Journal	www.linuxjournal.com
Online Linux	www.linux.org

Introduction

The Red Hat Linux operating system has become one of the major Linux distributions, bringing to the PC all the power and flexibility of a Unix workstation as well as a complete set of Internet applications and a fully functional desktop interface. This book is designed not only to be a complete reference on Red Hat Linux, but also to provide clear and detailed explanations of Linux features. No prior knowledge of Unix is assumed; Linux is an operating system anyone can use.

This book identifies seven major Linux topics: basic setup, environments and applications, the Internet, servers, administration, and network administration. These topics are integrated into the different ways Red Hat presents its distribution: as a desktop workstation, network workstation, server, and development platform. The section on the Red Hat Desktop workstation covers environments and applications. As a Red Hat network workstation, Internet applications are added. For Red Hat servers, the configuration and setup of various Internet servers are discussed. Included also are detailed sections on different system and network administration topics such as configuring the kernel, accessing files systems, and setting up firewalls. Though applicable primarily to programmers, the Red Hat development section introduces you to different development tools, such as compilers, shell scripting, and GUI construction. This section is available on the Web at **www.osborne.com**, should you wish to peruse this topic.

The first two sections of the book are designed to cover tasks you would need to perform to get your system up and running. They emphasize several of the key strengths provided by Red Hat Linux such as automatic update procedures, painless software installations, an extensive suite of software applications along with Red Hat specific configurations for them, and a collection of easy-to-use administrative tools for networking, user management, and server control. Red Hat distributions also include the most recent stable Linux software, such as security applications like Tripwire and IP-Tables, and the newest stable versions of the Gnome and KDE desktops, including the Nautilus file manager.

After an introduction to the working environment, including both Gnome and KDE desktops, you learn how to quickly update your system, access CD-ROMs, and set up your printer. The Red Hat Network makes software updates nearly automatic, letting you update the software on your system, including applications, all at once, with just a couple of mouse clicks. Internet access can be set up for modems, DSL, and Ethernet networks with easy-to-use GUI tools that guide you every step of the way. Security is a primary concern for any networked system. This section shows you how to implement basic protection methods such as encryption, intrusion detection, and firewalls. Many people now use Red Hat Linux to set up a home or local business network. The steps involved to implement a basic network can now be carried out using simple software tools. All these topics are covered in greater detail later in the book.

Gnome and the K Desktop Environment (KDE) have become standard desktop Graphical User Interfaces (GUI) for Linux, noted for their power, flexibility, and ease-of-use. These are complete desktop environments that are more flexible than either Windows or the Mac/OS. They support standard desktop features such as menus, taskbars, and drag-and-drop operations. But they also provide virtual desktops, panel applets and menus, and Internet-capable file managers. Gnome has become a standard GUI interface for Red Hat Linux systems, though Red Hat also provides full support for KDE, including it in its standard distribution. You can install both, run applications from one or the other, and easily switch from one to the other. Both Gnome and KDE were designed with software development in mind, providing a firm foundation that has encouraged the development of massive numbers of new applications for these interfaces. They have become integrated components of Linux, with applications and tools for every kind of task and operation. Instead of treating Gnome and KDE as separate entities, Gnome and KDE tools and applications are presented throughout the book. For example, Gnome and KDE mail clients are discussed along with other mail clients in Chapter 17. Gnome and KDE FTP clients, editors, graphic tools, administration tools, and others are also handled in their respective chapters (Chapter 5, 15, 16, 29, and 36).

Red Hat Linux is also a fully functional Unix operating system. It has all the standard features of a powerful Unix system, including a complete set of Unix shells such as BASH, TCSH, and the Z-shell. Those familiar with the Unix interface can use any of these shells, with the same Unix commands, filters, and configuration features.

For the Internet, Linux has become a platform for very powerful Internet applications. You are able not only to use the Internet, but also, with Linux, to become a part of it, creating your own Web, FTP, and Gopher sites. Other users can access your Linux systems, several at the same time, using different services. You can also use very powerful Gnome, KDE, and Unix clients for mail and news. Linux systems are not limited to the Internet. You can use them on any local intranet, setting up an FTP or Web site for your network. The Red Hat system provided on the CD-ROMs with this book come equipped with variety of fully functional FTP and Web servers already installed and ready to use. All you need to do is add the files you want onto your site.

Red Hat Linux has the same administration features found on standard Unix systems as well as several user-friendly GUI configuration tools that make any administration task a simple matter of choosing items on a menu or clicking a checkbox. It has the same multiuser and multitasking capabilities. You can set up accounts for different users, and each can access your system at the same time. Each user can have several programs running concurrently. With Linux you can control access, set up network connections, and install new devices. Red Hat Linux includes very powerful and easy-to-use, window-based configuration tools for tasks like configuring your printers and setting up your network connections. It is also compatible with comprehensive configuration tools like Linuxconf and Webmin.

A wide array of applications operates on Linux. Many personal versions of commercial applications are available for Linux free of charge, such as WordPerfect and Sybase databases. You can download them directly from the Internet. Numerous Gnome and KDE applications are continually released through their respective Web sites. The GNU public licensed software provides professional-level applications, such as programming development tools, editors and word processors, as well as numerous specialized applications, such as those for graphics and sound. A massive amount of software is available at online Linux sites where you can download applications and then easily install them onto your system.

What many may not realize is that all standard Linux systems support a wide range of programming languages, allowing users to create their own programs easily. All Red Hat Linux distributions include a large selection of programming platforms, including support for numerous kinds of shell programming, a variety of higher level languages like Perl and Gawk, and extensive GUI programming for desktops like Gnome and KDE. Once Red Hat Linux is installed, you can start creating your own programs.

Since this book is really six books in one-an Internet book, a Gnome and KDE book, a Server book, a Networking book, a Programming book, and an Administration book-how you choose to use it depends upon how you want to use your Red Hat Linux system. Almost all Linux operations can be carried out using either the Gnome or KDE interface. You need use the Unix command line interface very little, if at all. You can focus on the Gnome and KDE chapters and their corresponding tools and applications in the different chapters throughout the book. On the other hand, if you want to delve deeper into the Unix aspects of Linux, you can check out the Shell chapters and the corresponding shell-based applications in other chapters. If you want to use Red Hat Linux only for its Internet services, then concentrate on the Internet clients and servers, most of which are already installed for you. If you want to use Linux as a multiuser system servicing many users or integrate it into a local network, you can use the detailed system, file, and network administration information provided in the administration chapters. None of these tasks are in any way exclusive. If you are working in a business environment, you will probably make use of all three aspects. Single users may concentrate more on the desktops and the Internet features, whereas administrators may make more use of the Unix features.

Part I provides an introduction to Red Hat Linux along with a listing of Red Hat Linux resources, including software repositories, documentation sites, newsgroups, and Linux news and development sites. This part also covers the streamlined installation procedure for Red Hat, which takes about 30 minutes or less. The Red Hat installation tool provides excellent commentary, describing each step in detail. In this section, you also learn the essentials of using both Gnome and KDE, along with the basics of working on the shell command line.

Part II is designed to help you start using Linux quickly. System configuration tasks, such as mounting CD-ROMs and adding new user accounts, are presented using the easiest methods, without the complex detail described in the administration chapters. Basic network configuration tasks are discussed, such as setting up a connection to an Internet Service Provider (ISP) over a modem and entering ISP network information such as nameserver and gateway addresses. Then, a brief discussion of network security methods shows you how to quickly set up simple protection for a single system connected to the Internet, as well as where to find out more if you have more complex security requirements. Finally, you learn how to set up a small local network that could even include Windows systems. You see how each host should be connected and configured, and how to create a gateway, connecting local hosts to the Internet. You learn to quickly configure and run services, such as the Domain Name Service, Samba, Sendmail, and a Web site.

Part III of this book deals with Red Hat Linux as a Desktop Workstation. Here you are introduced to the different kinds of user environments and applications available for Linux, starting with KDE and Gnome. Different features such as applets, the Panel, and configuration tools are described in detail. Using either of these interfaces, you can run all your applications using icons, menus, and windows. Plus, at any time, you can open up a terminal window through which you can enter standard Linux commands on a command line. The new Nautilus file manager for Gnome is covered in detail. You can also use the standard Unix command-line interface to run any of the standard Unix commands. Next, the BASH shell and its various file, directory, and filter commands are examined. The remaining chapters in this section discuss the applications available for Linux, beginning with Office suites such as KOffice and Star Office. KOffice is now distributed as part of KDE and can be found on most Linux distributions. The different database management systems available are also discussed along with the Web site locations where you can download them. A variety of different text editors is also available, including several Gnome and KDE editors, as well as the Vim (enhanced VI), gvim (graphical Vi), and GNU Emacs editors.

Part IV of this book incorporates the added features of Red Linux as a network workstation. Here the book discusses in detail the many Internet applications you can use on your Linux system. Red Hat Linux automatically installs mail, news, FTP, and Web browser applications, as well as FTP and Web servers. Both KDE and Gnome come with a full set of mail, news, FTP clients, and Web browsers. These are described in detail along with Netscape communicator, now an integrated part of all Linux systems. On your CD-ROMs, there are other mail clients, newsreaders, and Internet tools that you can easily install from your desktop. In addition, the book describes Internet clients, such as Balsa, that you can download from Internet sites and install on your system.

Part V discusses Internet servers you can run on Red Hat, including FTP, Web, and DNS servers. Internet servers have become integrated components of most Linux systems. Both the standard wu-ftpd FTP server and the newer ProFTPD server with its directive format are presented. ProFTPD covers features like guest and virtual FTP sites. The Apache Web Server chapter covers standard configuration directives such as those for automatic indexing as well as the newer virtual host directives. Apache GUI configuration tools, such as comanche, are also presented. Configuration files and features for the Domain Name System (DNS) and its BIND server are examined in detail along with features like virtual domains and IP aliases.

With Linux, you can easily set up your own Domain Name Server for a home or small local network. Both sendmail and POP mail servers are covered. The INN news server, the Squid proxy server, and the ht:/DIG and WAIS search servers are also examined.

Part VI discusses system administration topics including user, software, file system, device, kernel, and X Window administration. These chapters emphasize the use of GUI system management configuration tools available on Red Hat Linux. There are also detailed descriptions of the configuration files used in administration tasks and how to make entries in them. First, basic system administration tasks are covered, such as selecting runlevels, monitoring your system, and scheduling shutdowns. Then aspects of setting up and controlling users and groups are discussed. Presentations include both the GUI tools you can use for these tasks and the underlying configuration files and commands. Software installation has been simplified with the Package Management System (RPMS). There are GUI tools like GnomeRPM that you can use to easily install and uninstall software, much as you would with the Windows Install Wizard. Different file system tasks are covered-mounting file systems, selecting device names, and accessing Windows files. Device configuration covers topics such as device files, installing printers, and using the kernel modules to support new devices. Using, updating, and configuring the Linux kernel with its modules is covered in detail along with procedures for installing new kernels. X Window system topics cover the XFree86 servers, window manager configuration, X Window system startup methods, such as the display manger, and X Window system configuration commands.

Part VII covers network administration, dealing with topics such as configuring remote file system access and setting up firewalls. Most network administration tasks can be performed using Red Hat GUI configuration tools like netcfg. These are discussed in detail first. Next, the various network file system interfaces and services, such as NFS for Unix, NIS, and NetaTALK for AppleTalk networks, are presented. The next chapter on Samba shows how to access Windows file systems and printers. Then the different aspects of network administration are discussed, such as network connections and routers, Domain Name Service, hostname designations, IP virtual hosts, and IP masquerading. Network security topics cover firewalls and encryption using netfilter (iptables) to protect your system, the Secure Shell (SSH) to provide secure remote transmissions, and Kerberos to provide secure authentication (the older ipchains firewall system is also covered).

On the Web at **www.osborne.com**, you'll find several chapters that discuss basic components used for development on Red Hat systems, beginning with shell programming for the BASH shell where you can create complex shell scripts. Compilers, libraries, and programming tools are then covered, including such topics as the GCC compiler, managing static and shared libraries, and using development tools like make, the gdb debugger, and CVS revision manager. Then both the development tools for KDE and Gnome programming are covered, including KDevelop and Glade. Finally, a brief survey of Perl, Tcl/TK, and Gawk shows you how to easily construct programs with powerful file-management capabilities, string manipulation, and control structures, as well as GUI interfaces.

Part I: Introduction

Chapter List

Chapter 1: Introduction to Red Hat Linux

Overview

Linux is an operating system for PC computers and workstations that now features a fully functional graphical user interface (GUI), just like Windows and the Mac (though more stable). Linux was developed in the early 1990s by Linus Torvald, along with other programmers around the world. As an operating system, Linux performs many of the same functions as UNIX, Mac, Windows, and Windows NT. However, Linux is distinguished by its power and flexibility. Most PC operating systems, such as Windows, began their development within the confines of small, restricted personal computers, which have only recently become more versatile machines. Such operating systems are constantly being upgraded to keep up with the ever-changing capabilities of PC hardware. Linux, on the other hand, was developed in a different context. Linux is a PC version of the UNIX operating system that has been used for decades on mainframes and minicomputers, and is currently the system of choice for workstations. Linux brings the speed, efficiency, and flexibility of UNIX to your PC, taking advantage of all the capabilities that personal computers can now provide. Along with its UNIX capabilities come powerful networking features, including support for Internet, intranets, Windows, and AppleTalk networking. As a standard, Linux is distributed with fast, efficient, and stable Internet servers, such as the Web, FTP, and Gopher servers, along with domain name, proxy, news, mail, and indexing servers. In other words, Linux has everything you need to set up, support, and maintain a fully functional network.

Now, with both Gnome and K Desktop, Linux also provides GUI interfaces with that same level of flexibility and power. Unlike Windows and the Mac, you can choose the interface you want and then customize it further, adding panels, applets, virtual desktops, and menus, all with full drag-and-drop capabilities and Internet-aware tools. On your desktop, a file manager window can access any Internet site, enabling you to display Web pages and download files with a few simple mouse operations. To print a file, simply drag it to a Printer icon.

Linux does all this at a great price. Linux is free, including the network servers and GUI desktops. Unlike the official UNIX operating system, Linux is distributed freely under a GNU General Public License as specified by the Free Software Foundation, making it available to anyone who wants to use it. GNU stands for "Gnu's Not UNIX" and is a project initiated and managed by the Free Software Foundation to provide free software to users, programmers, and developers. Linux is copyrighted, and it is not public domain. However, a GNU public license has much the same effect as being in the public domain. The GNU public license is designed to ensure Linux remains free and, at the same time, standardized. Only one official Linux exists. Linux is technically the operating system kernel, the core operations. In addition Linux is commonly bundled with an extensive set of software available under the GNU public license, including environments, programming languages, Internet tools, and text editors. People sometimes have the mistaken impression that Linux is somehow less than a professional operating system because it is free. Linux is, in fact, a PC and workstation version of UNIX. Many consider it far more stable and much more powerful than Windows. This power and stability have made Linux an operating system of choice as a network server.

To appreciate Linux completely, you need to understand the special context in which the UNIX operating system was developed. UNIX, unlike most other operating systems, was developed in a research and academic environment. In universities, research laboratories, data centers, and enterprises, UNIX is the system of choice. Its development paralleled the entire computer and communications revolution over the past several decades. Computer professionals often developed new computer technologies on UNIX, such as those developed for the Internet. Although a sophisticated system, UNIX was designed from the beginning to be flexible. The UNIX system itself can be easily modified to create different versions. In fact, many different vendors maintain different official versions of UNIX. IBM, Sun, and Hewlett-Packard all sell and maintain their own versions of UNIX. The unique demands of research programs often require that UNIX be tailored to their own special needs. This inherent flexibility in the UNIX design in no way detracts from its quality. In fact, this flexibility attests to the ruggedness of UNIX, allowing it to adapt to practically any environment. This is the context in which Linux was developed. Linux is, in this sense, one other version of UNIX-a version for the PC. The development of Linux by computer professionals working in a research-like environment reflects the way UNIX versions have usually been developed. Linux is publicly licensed and free-and reflects the deep roots UNIX has in academic institutions, with their sense of public service and support. Linux is a top-rate operating system accessible to everyone, free of charge.

As a way of introducing Linux, this chapter discusses Linux as an operating system, the history of Linux and UNIX, the overall design of Linux, and Linux distributions. This chapter also discusses online resources for documentation, software, and newsgroups, plus Web sites with the latest news and articles on Linux. Web and FTP site listings are placed in tables at the end of this chapter for easy reference. Here you can find sites for different distributions, Linux publications, software repositories, and Linux development, as well as for office suites and commercial databases.

Red Hat Linux

Red Hat Linux is currently the most popular Linux distribution. As a company, Red Hat provides software and services to implement and support professional and commercial Linux systems. Red Hat freely distributes its version of Linux under the GNU Public License. Red Hat generates income by providing professional-level support, consulting, and training services. The Red Hat Certified Engineers (RHCE) training and certification program is designed to provided reliable and highly capable administrators and developers to maintain and customize professional-level Red Hat systems. Red Hat has forged software alliances with major companies like Oracle, IBM, and Sun, often bundling applications such as IBM's DB2 database system with Red Hat Linux. Red Hat also maintains a strong commitment to open source Linux applications. Red Hat originated the RPM package system used on several distributions, which automatically installs and removes software packages. Red Hat is also providing much of the software development for the Gnome desktop, and it is a strong supporter of KDE. Its distribution includes both Gnome and KDE.

The Red Hat distribution of Linux is available online at numerous FTP sites. It maintains its own FTP site at **ftp.redhat.com**, where you can download the entire current release of Red Hat Linux, as well as updates and third-party software. Red Hat was designed from its inception to work on numerous hardware platforms. Currently, Red Hat supports Sparc, Intel, and Alpha platforms. See **www.redhat.com** for more information, including extensive documentation such as Red Hat manuals, FAQs, and links to other Linux sites.

If you purchase Red Hat Linux from Red Hat, you are entitled to online support services. Although Linux is free, Red Hat as a company specializes in support services, providing customers with its expertise in developing solutions to problems that may arise or using Linux to perform any of several possible tasks, such as e-commerce or database operations.

Red Hat maintains an extensive library of Linux documentation that is freely accessible online (see Table 1-1). On its Web page, you can link to its support page, which lists the complete set of Red Hat manuals, all in Web page format for easy viewing with any Web browser. These include the Reference Guide, the Getting Started Guide, and the Installation Guide. Tip, HOW-TO, and FAQ documents are also provided. Of particular note is the Hardware Compatibility Lists. This documentation lists all the hardware compatible with Red Hat Linux. For PC users, this includes most hardware, with few exceptions. All the Red Hat documentation is freely available under the GNU Public License.

Table 1-1: Red Hat Linux Resources

References	**Description**
www.redhat.com	Red Hat Web site
www.redhat.com/support	Support page for Red Hat Linux, including links to current online documentation
The Official Red Hat Linux Getting Started Guide	A getting started guide for first time users
The Official Red Hat Linux Installation Guide	Detailed installation guide for Red Hat Linux
Red Hat Linux Installation Gotchas	Installation troubleshooting
Red Hat Reference Guide	The Red Hat Reference
Red Hat Customization Guide	Topics covering common customization tasks and tools, such as server configurations, GnomeRPM package manager, and Linuxconf administration tool
Red Hat Linux FAQ	Frequently asked questions for Red Hat Linux
Introduction to Linux	Frequently asked questions on Linux

Before installing Red Hat Linux on your system, you should check the Installation guide. This is a lengthy and detailed document that takes you through each step of the process carefully. If your system is designed for any special tasks, be sure to consult the Customization guide. This document covers a vriety of topics such as automatic installation on networks using Red Hat kickstart; network services like Samba, Apache, and FTP; system administration tools; and software package installation and management.

Red Hat also provides documentation on implementing PPP Internet connections, Samba file sharing, Apache Web server, firewalls, mail servers, and for the Credit Card Verification System (CCVS), a Red Hat commercial product.

Operating Systems and Linux

An *operating system* is a program that manages computer hardware and software for the user. Operating systems were originally designed to perform repetitive hardware tasks. These tasks centered around managing files, running programs, and receiving commands from the user. You interact with an operating system through a user interface. This user interface allows the operating system to receive and interpret instructions sent by the user. You only need to send an instruction to the operating system to perform a task, such as reading a file or printing a document. An operating system's user interface can be as simple as entering commands on a line or as complex as selecting menus and icons on a desktop.

An operating system also manages software applications. To perform different tasks, such as editing documents or performing calculations, you need specific software applications. An *editor* is an example of a software application that enables you to edit a document, making changes and adding new text. The editor itself is a program consisting of instructions to be executed by the computer. To use the program, it must first be loaded into computer memory, and then its instructions are executed. The operating system controls the loading and execution of all programs, including any software applications. When you want to use an editor, simply instruct the operating system to load the editor application and execute it.

File management, program management, and user interaction are traditional features common to all operating systems. Linux, like all versions of UNIX, adds two more features. Linux is a multiuser and multitasking system. In a multitasking system, you can ask the system to perform several tasks at the same time. While one task is being done, you can work on another. For example, you can edit a file while another file is being printed. You do not have to wait for the other file to finish printing before you edit. In a multiuser system, several users can log in to the system at the same time, each interacting with the system through his or her own terminal.

Operating systems were originally designed to support hardware efficiency. When computers were first developed, their capabilities were limited and the operating system had to make the most of them. In this respect, operating systems were designed with the hardware in mind, not the user. Operating systems tended to be rigid and inflexible, forcing the user to conform to the demands of hardware efficiency.

Linux, on the other hand, is designed to be flexible, reflecting its UNIX roots. As a version of UNIX, Linux shares the same flexibility designed for UNIX, a flexibility stemming from UNIX's research origins. The UNIX operating system was developed by Ken Thompson at AT&T Bell Laboratories in the late 1960s and early 1970s. The UNIX system incorporated many new developments in operating system design. Originally, UNIX was designed as an operating system for researchers. One major goal was to create a system that could support the researchers' changing demands. To do this, Thompson had to design a system that could deal with many different kinds of tasks. Flexibility became more important than hardware efficiency. Like UNIX, Linux has the advantage of being able to deal with the variety of tasks any user may face.

This flexibility allows Linux to be an operating system that is accessible to the user. The user is not confined to limited and rigid interactions with the operating system. Instead, the operating system is thought of as providing a set of highly effective tools available to the user.

This user-oriented philosophy means you can configure and program the system to meet your specific needs. With Linux, the operating system becomes an operating environment.

History of Linux and UNIX

Since Linux is a version of UNIX, its history naturally begins with UNIX. The story begins in the late 1960s when a concerted effort to develop new operating system techniques occurred. In 1968, a consortium of researchers from General Electric, AT&T Bell Laboratories, and the Massachusetts Institute of Technology carried out a special operating system research project called MULTICS (MULTiplexed Information Computing System). MULTICS incorporated many new concepts in multitasking, file management, and user interaction. In 1969, Ken Thompson, Dennis Ritchie, and the researchers at AT&T Bell Laboratories developed the UNIX operating system, incorporating many of the features of the MULTICS research project. They tailored the system for the needs of a research environment, designing it to run on minicomputers. From its inception, UNIX was an affordable and efficient multiuser and multitasking operating system.

The UNIX system became popular at Bell Labs as more and more researchers started using the system. In 1973, Dennis Ritchie collaborated with Ken Thompson to rewrite the programming code for the UNIX system in the C programming language. Dennis Ritchie, a fellow researcher at Bell Labs, developed the C programming language as a flexible tool for program development. One of the advantages of C is it can directly access the hardware architecture of a computer with a generalized set of programming commands. Up until this time, an operating system had to be specially rewritten in a hardware-specific assembly language for each type of computer. The C programming language allowed Dennis Ritchie and Ken Thompson to write only one version of the UNIX operating system, which could then be compiled by C compilers on different computers. In effect, the UNIX operating system became transportable, able to run on a variety of different computers with little or no reprogramming.

UNIX gradually grew from one person's tailored design to a standard software product distributed by many different vendors, such as Novell and IBM. Initially, UNIX was treated as a research product. The first versions of UNIX were distributed free to the computer science departments of many noted universities. Throughout the 1970s, Bell Labs began issuing official versions of UNIX and licensing the systems to different users. One of these users was the Computer Science department of the University of California, Berkeley. Berkeley added many new features to the system that later became standard. In 1975, Berkeley released its own version of UNIX, known by its distribution arm, Berkeley Software Distribution (BSD). This BSD version of UNIX became a major contender to the AT&T Bell Labs version. Other independently developed versions of UNIX sprouted up. In 1980, Microsoft developed a PC version of UNIX called Xenix. AT&T developed several research versions of UNIX and, in 1983, it released the first commercial version, called System 3. This was later followed by System V, which became a supported commercial software product. You can find more information on UNIX in *UNIX: The Complete Reference*, written by the UNIX experts at AT&T labs, Kenneth Rosen, Doug Host, James Farber, and Richard Rosinski.

At the same time, the BSD version of UNIX was developing through several releases. In the late 1970s, BSD UNIX became the basis of a research project by the Department of Defense's Advanced Research Projects Agency (DARPA). As a result, in 1983, Berkeley released a powerful version of UNIX called BSD release 4.2. This release included sophisticated file management as well as networking features based on TCP/IP network protocols-the same protocols now used for the Internet. BSD release 4.2 was widely distributed and adopted by many vendors, such as Sun Microsystems.

The proliferation of different versions of UNIX led to a need for a UNIX standard. Software developers had no way of knowing on what versions of UNIX their programs would actually run. In the mid-1980s, two competing standards emerged, one based on the AT&T version of UNIX and the other based on the BSD version. In bookstores today, you can find many different books on UNIX for one or the other version. Some specify System V UNIX, while others focus on BSD UNIX.

AT&T moved UNIX to a new organization, called UNIX System Laboratories, which could focus on developing a standard system, integrating the different major versions of UNIX. In 1991, UNIX System Laboratories developed System V release 4, which incorporated almost all the features found in System V release 3, BSD release 4.3, SunOS, and Xenix. In response to System V release 4, several other companies, such as IBM and Hewlett-Packard, established the Open Software Foundation (OSF) to create their own standard version of UNIX. Two commercial standard versions of UNIX existed then-the OSF version and System V release 4. In 1993, AT&T sold off its interest in UNIX to Novell. UNIX Systems Laboratories became part of Novell's UNIX Systems Group. Novell issued its own versions of UNIX based on System V release 4, called UNIXWare, designed to interact with Novell's NetWare system. UNIX Systems Laboratories is currently owned by the Santa Cruz Operation. With Solaris, Sun has introduced System V release 4 onto its Sun systems. Two competing GUIs for UNIX, called Motif and OpenLook, have been superseded by a new desktop standard called the Common Desktop Environment (CDE), which has since been incorporated into OpenMotif, an open source version of Motif also for use on Linux.

Throughout much of its development, UNIX remained a large and demanding operating system requiring a workstation or minicomputer to be effective. Several versions of UNIX were designed primarily for the workstation environment. SunOS was developed for Sun workstations and AIX was designed for IBM workstations. As personal computers became more powerful, however, efforts were made to develop a PC version of UNIX. Xenix and System V/386 are commercial versions of UNIX designed for IBM-compatible PCs. AUX is a UNIX version that runs on the Macintosh. A testament to UNIX's inherent portability is that it can be found on almost any type of computer: workstations, minicomputers, and even supercomputers. This inherent portability made possible an effective PC version of UNIX.

Linux was originally designed specifically for Intel-based personal computers. Linux started out as a personal project of a computer science student named Linus Torvald at the University of Helsinki. At that time, students were making use of a program called *Minix,* which highlighted different UNIX features. Minix was created by Professor Andrew Tannebaum and widely distributed over the Internet to students around the world. Linus's intention was to create an effective PC version of UNIX for Minix users. He called it Linux, and in 1991, Linus released version 0.11. Linux was widely distributed over the Internet and, in the following years, other programmers refined and added to it, incorporating most of the applications and features now found in standard UNIX systems. All the major window managers have been ported to Linux. Linux has all the Internet utilities, such as FTP file transfer support, Web browsers, and remote connections with PPP. It also has a full set of program development utilities, such as C++ compilers and debuggers. Given all its features, the Linux operating system remains small, stable, and fast. In its simplest format, Linux can run effectively on only 2MB of memory.

Although Linux has developed in the free and open environment of the Internet, it adheres to official UNIX standards. Because of the proliferation of UNIX versions in the previous decades, the Institute of Electrical and Electronics Engineers (IEEE) developed an independent UNIX standard for the American National Standards Institute (ANSI). This new ANSI-standard UNIX is called the Portable Operating System Interface for Computer Environments (POSIX). The standard defines how a UNIX-like system needs to operate, specifying details such as system calls and interfaces. POSIX defines a universal standard to which all UNIX versions must adhere. Most popular versions of UNIX are now POSIX-compliant. Linux was developed from the beginning according to the POSIX standard. Linux also adheres to the Linux file system standard (FSSTND), which specifies the location of files and directories in the Linux file structure. See **www.pathname.com/fhs** for more details.

Linux Overview

Like UNIX, Linux can be generally divided into three major components: the kernel, the environment, and the file structure. The *kernel* is the core program that runs programs and manages hardware devices, such as disks and printers. The *environment* provides an interface for the user. It receives commands from the user and sends those commands to the kernel for execution. The *file structure* organizes the way files are stored on a storage device, such as a disk. Files are organized into directories. Each directory may contain any number of subdirectories, each holding files. Together, the kernel, the environment, and the file structure form the basic operating system structure. With these three, you can run programs, manage files, and interact with the system.

An environment provides an interface between the kernel and the user. It can be described as an interpreter. Such an interface interprets commands entered by the user and sends them to the kernel. Linux provides several kinds of environments: desktops, window managers, and command line shells. Each user on a Linux system has his or her own user interface. Users can tailor their environments to their own special needs, whether they be shells, window managers, or desktops. In this sense, for the user, the operating system functions more as an operating environment, which the user can control.

The shell interface is simple and usually consists of a prompt at which you type a command, and then press ENTER. In a sense, you are typing the command on a line; this line is often referred to as the *command line*. You will find that the commands entered on the command line can become quite complex. Over the years, several different kinds of shells have been developed and, currently, three major shells exist: Bourne, Korn, and C shell. The *Bourne shell* was developed at Bell Labs for System V. The *C shell* was developed for the BSD version of UNIX. The *Korn shell* is a further enhancement of the Bourne shell. Current versions of UNIX, including Linux, incorporate all three shells, enabling you to choose the one you prefer. However, Linux uses enhanced or public domain versions of these shells: the Bourne Again shell, the TC shell, and the Public Domain Korn shell. When you start your Linux system, you are placed in the Bourne Again shell, an updated version of the Bourne shell. From there, you can switch to other shells as you choose.

As an alternative to a command line interface, Linux provides both desktops and window managers. These use GUIs based on the X Window System developed for UNIX by the Open

Group consortium (**www.opengroup.org**). A *window manager* is a reduced version of a integrated desktop, supporting only window operation, but it still enables you to run any application. A desktop provides a complete GUI, much like Windows and the Mac. You have windows, icons, and menus, all managed through mouse controls. Currently, two desktops are freely available and both are included with most distributions of Linux: Gnome and KDE.

In Linux, files are organized into directories, much as they are in Windows. The entire Linux file system is one large interconnected set of directories, each containing files. Some directories are standard directories reserved for system use. You can create your own directories for your own files, as well as easily move files from one directory to another. You can even move entire directories, and share directories and files with other users on your system. With Linux, you can also set permissions on directories and files, allowing others to access them or restricting access to you alone. The directories of each user are, in fact, ultimately connected to the directories of other users. Directories are organized into a hierarchical tree structure, beginning with an initial root directory. All other directories are ultimately derived from this first root directory.

Desktops

With the K Desktop Environment (KDE) and the GNU Network Object Model Environment (Gnome), Linux now has a completely integrated GUI interface. You can perform all your Linux operations entirely from either interface. Previously, Linux did support window managers that provided some GUI functionality, but they were usually restricted to window operations. KDE and Gnome are fully operational desktops supporting drag-and-drop operations, enabling you to drag icons to your desktop and to set up your own menus on an Applications panel. Both rely on an underlying X Window System, which means as long as they are both installed on your system, applications from one can run on the other desktop. You can run KDE programs like the KDE mailer or the newsreader on the Gnome desktop. Gnome applications like the Gftp FTP client can run on the KDE desktop. You can even switch file managers, running the KDE file manager on Gnome. You lose some desktop functionality, such as drag-and- drop operations, but the applications run fine. Desktop and window manager sites are listed in Table 1-2. The Gnome and KDE sites are particularly helpful for documentation, news, and software you can download for those desktops.

Both desktops can run any X Window System program, as well as any cursor-based program like Emacs and Vi, which were designed to work in a shell environment. At the same time, a great many applications are written just for those desktops and included with your distributions. The K Desktop has a complete set of Internet tools, along with editors and graphic, multimedia, and system applications. Gnome has slightly fewer applications, but a great many are currently in the works. Check their Web sites at **www.gnome.org** and **www.kde.org** for new applications. As new versions are released, they include new software.

Note Ximian currently maintains an enhanced version of Gnome called Ximian Gnome at **www.ximian.com**.

Table 1-2: Desktops and Window Managers

URL	Internet Site
www.gnome.org	Gnome Web site
www.kde.org	K Desktop Environment Web site

Table 1-2: Desktops and Window Managers

URL	Internet Site
www.x11.org	X Window System Web site, with links
www.fvwm.org	FVWM window manager
www.windowmaker.org	WindowMaker window manager
www.enlightenment.org	Enlightenment window manager
www.afterstep.org	AfterStep window manager
www.blackbox.org	Blackbox window manager
www.lesstif.org	Hungry Programmers OSF/Motif
www.themes.org	Desktop and Window manager themes, including KDE and Gnome
www.xfree86.org	XFree86, GNU version of the X Window System provided for Linux
www.themes.org	Themes for window mangers and desktops
www.Ximian.com	Ximian Gnome
www.openmotif.org	OpenMotif, open source version of Motif

Open Source Software

Linux was developed as a cooperative effort over the Internet, so no company or institution controls Linux. Software developed for Linux reflects this background. Development often takes place when Linux users decide to work on a project together. When completed, the software is posted at an Internet site, and any Linux user can then access the site and download the software. The potential for Linux-based software is explosive. Linux software development has always operated in an Internet environment and it is global in scope, enlisting programmers from around the world. The only thing you need to start a Linux-based software project is a Web site.

Most Linux software is developed as open source software. This means that the source code for an application is freely distributed along with the application. Programmers over the Internet can make their own contributions to a software's development, modifying and correcting the source code. Linux is considered open source. Its source code in included in all its distributions and is freely available on the Internet. Many major software development efforts are also open source projects such as the KDE and Gnome desktops along with most of their applications. Netscape Communicator Web browser package has also become open source, with all its source code freely available. The OpenOffice office suite supported by Sun is an open source project based on the former StarOffice office suite. Recently much of the Tcl/TK development tools have become open source projects. Many of the open source applications that run on Linux have located their Web sites at Source Forge (sourceforge.net). Source Forge is a hosting site designed specifically to support open source projects. You can find more information about the open source movement and recent developments at both Linuxcare (**www.linuxcare.com**) and at **www.opensource.org**. Red Hat also hosts open source projects at **sources.redhat.com**.

Open source software is protected by public licenses. These prevent commercial companies from taking control of open source software by adding a few modifications of their own, copyrighting those changes, and selling the software as their own product. The most popular public license is the GNU Public License provided by the Free Software Foundation. This is the license that Linux is distributed under. The GNU Public License retains copyright, freely licensing the software with the requirement that the software and any modifications made to it are always freely available. Other public licenses have also been created to support the demands of different kinds of open source project. The Lesser GNU Public License (LGPL) lets commercial applications use GNU licensed software libraries. Netscape made its Netscape Communicator software available under a Netscape Public License (NPL) that covers modifications made directly to the Netscape source code. Additions made to Netscape are covered under the Mozilla Public License. The QT Public License (QPL) lets open source developers use the QT libraries essential to the KDE desktop. You can find a complete listing at **www.opensource.org**.

Linux is currently copyrighted under a GNU public license provided by the Free Software Foundation, and is often referred to as GNU software (see **www.gnu.org**). GNU software is distributed free, provided it is freely distributed to others. GNU software has proven both reliable and effective. Many of the popular Linux utilities, such as C compilers, shells, and editors, are all GNU software applications. Installed with your Linux distribution are the GNU C++ and Lisp compilers, Vi and Emacs editors, BASH and TCSH shells, as well as Tax and Ghostscript document formatters. In addition there are many open source software projects that are licensed under the GNU Public License (GPL). Many of these software applications are available at different Internet sites, and these are listed in Table 1-3. Chapter 4 and Chapter 31 describe in detail the process of downloading software applications from Internet sites and installing them on your system.

Under the terms of the GNU General Public License, the original author retains the copyright, although anyone can modify the software and redistribute it, provided the source code is included. Also, no restriction exists on selling the software or giving it away free. One distributor could charge for the software, while another one could provide it free of charge.

Lately, major software companies are also developing Linux versions of their most popular applications. A Linux version of Sun's Java Software Development Kit (SDK) is also available through **www.blackdown.org**. Corel has developed a Linux version of WordPerfect, while Oracle provides a Linux version of its Oracle database. (At present, no plans seem in the works for Microsoft applications.) Until recently, however, many of these lacked a true desktop interface, but this has changed dramatically with the introduction of KDE and Gnome. These desktops are not merely interfaces: They both provide extensive, flexible, and powerful development libraries that software developers can use to create almost any kind of application, which they are.

Linux Software

A great deal of Linux software is currently available from online sources. You can download applications for desktops, Internet servers, office suites, and programming packages, among others. Several centralized repositories make it easy to locate an application and find information about it. Of particular note are **sourceforge.net**, **freshmeat.net**, **rpmfind.net**, **apps.kde.com**, and **linuxapps.com**.

Software packages are distributed either in compressed archives or in RPM packages. RPM packages are those archived using the Red Hat Package Manager. Compressed archives have an extension such as **.tar.gz** or **.tar.Z**, whereas RPM packages have an **.rpm** extension. For Red Hat, downloading the RPM package versions of software from their FTP sites is best. Whenever possible, you should try to download software from a distribution's FTP site, but you could also download the source version and compile it directly on your system. This has become a simple process, almost as simple as installing the compiled versions (see Chapter 4).

Red Hat also has a large number of mirror sites from which you can download their software packages. Red Hat mirror sites are listed at **www.redhat.com/download/mirror.html**. Most Linux Internet sites that provide extensive software archives have mirror sites, such as **www.kernel.org**, that hold the new Linux kernels. If you have trouble connecting to a main FTP site, try one of its mirrors.

The following tables list different sites for Linux software. Repositories and archives for Linux software are listed in Table 1-3, along with several specialized sites, such as those for commercial and game software. When downloading software packages, always check to see if versions are packaged for your particular distribution. For example, Red Hat will use RPM packages. Many sites provide packages for the different popular distributions, such as Red Hat, Caldera, and Debian. For others, first check the distribution FTP sites for a particular package. For example, a Red Hat package version for ProFTPD is located at the **ftp.redhat.com** FTP site. **rpmfind.net**, **freshmeat.net**, **sourceforge.net**, and **www.linuxapps.com** are also good places for locating RPM packages for particular distributions.

Table 1-3: Linux Software Archives, Repositories, and Links

URL	**Internet Site**
www.linuxapps.com	Linux Software Repository
sourceforge.net	Source Forge, open source software development sites for Linux applications and software repository
www.happypenguin.org/	Linux Game Tome
www.linuxgames.org	Linux games
www.linuxquake.com	Quake
http://www.xnet.com/~blatura/linapps.shtml	Linux applications and utilities page
freshmeat.net	New Linux software
www.linuxlinks.com	Linux links
filewatcher.org	Linux FTP site watcher
www.linuxdoc.org/links.html	Linux links
rpmfind.net	RPM package repository
www.gnu.org	GNU archive
www.opensound.com	Open sound system drivers
www.blackdown.org	Web site for Linux Java
www.fokus.gmd.de/linux	Woven goods for Linux

Table 1-3: Linux Software Archives, Repositories, and Links

URL	Internet Site
metalab.unc.edu	Mirror site for Linux software and distributions
www.linux.com	Linux software
sources.redhat.com	Open-source software hosted by Red Hat

Linux Office and Database Software

Many professional-level databases and office suites are now available for Linux. These include Oracle and IBM databases as well as the OpenOffice and K Office office suites. Table 1-4 lists sites for office suites and databases. Many of these sites provide free personal versions of their software for Linux, and others are entirely free. You can download from them directly and install on your Linux system.

Table 1-4: Database and Office Software

URL	Databases
www.oracle.com	Oracle database
www.sybase.com	Sybase database
www.software.ibm.com/data/db2/linux	IBM database
www.informix.com/linux	Informix database
www.cai.com/products/ingres.htm	Ingress II
www.softwareag.com	Adabas D database
www.mysql.com	MySQL database
www.ispras.ru/~kml/gss	The GNU SQL database
www.postgresql.org	The PostgreSQL database
www.fship.com/free.html	Flagship (Interface for xBase database files)
koffice.kde.org	Katabase (KOffice desktop database)
gaby.netpedia.net	Gaby (Gnome desktop personal database)
Office Software:	
koffice.kde.org	Koffice
linux.corel.com	WordPerfect
www.sun.com/staroffice	StarOffice
www.openoffice.org	Open Office
www.gnome.org/gw.html	Gnome Workshop Project
www.redhat.com	Applixware (commercial)

Internet Servers

One of the most important features of Linux, as well as all UNIX systems, is its set of Internet clients and servers. The Internet was designed and developed on UNIX systems, and Internet clients and servers, such as those for FTP and the Web, were first implemented on BSD versions of UNIX. DARPANET, the precursor to the Internet, was set up to link UNIX systems at different universities across the nation. Linux contains a full set of Internet clients and servers including mail, news, FTP, and Web, as well as proxy clients and servers. Sites for Internet server software available for Linux are listed in Table 1-5. Most of these are already included on the Red Hat CD-ROM included with this book; however, you can obtain news, documentation, and recent releases directly from the server's Web sites.

Table 1-5: Network Servers and Security

URL	Servers
www.apache.org	Apache Web server
www.proftpd.org	ProFTPD FTP server
www.isc.org	Internet Software Consortium: BIND, INN, and DHCPD
www.sendmail.org	Sendmail mail server
www.squid.org	Squid proxy server
www.samba.org	Samba SMB (Windows network) sever
boombox.micro.umn.edu/pub/gopher	Gopher server
www.eudora.com/free/qpop.html	Qpopper POP3 mail server
Netfilter.kernelnotes.org	IP Tables firewall server
netfilter.kernelnotes.org/ipchains	IP Chains firewall server
www.ssh.com	Secure Shell encryption
http://web.mit.edu/kerberos/www	Kerberos network authentication protocol
www.openssh.com	Open Secure Shell (free version of SSH)

Development Resources

Linux has always provided strong support for programming languages and tools. All distributions include the GNU C and C++ compiler (gcc) with supporting tools like make. Most distributions come with full development support for the KDE and Gnome desktops, letting you create your own Gnome and KDE applications. You can also download the Linux version of the Java Software Development Kit for creating Java programs. Perl and Tcl/TK versions of Linux are also included with most distributions. You can download current versions from their Web sites. Table 1-6 lists different sites of interest for Linux programming.

Table 1-6: Linux Programming

URL	Internet Sites
www.linuxprogramming.org	Linux programming resources
www.gnu.org	Linux compilers and tools (gcc).

Table 1-6: Linux Programming

URL	Internet Sites
dev.scriptics.com	Tcl Developers Xchange, Tk/Tcl products
java.sun.com	Sun Java Web site
www.perl.com	Perl Web site with Perl software
www.blackdown.org	Sun's Java Software Development Kit for Linux
developer.gnome.org	Gnome developers Web site
www.openprojects.nu	Open Projects Network
developer.kde.org	Developer's library for KDE
www.linuxcare.org	Linux open source software support

Online Information Sources

Extensive online resources are available on almost any Linux topic. The tables in this chapter list sites where you can obtain software, display documentation, and read articles on the latest developments. Many Linux Web sites provide news, articles, and information about Linux. Several are based on popular Linux magazines, such as **www.linuxjournal.com** and **www.linuzgazzette.com**. Others operate as Web portals for Linux, such as **www.linux.com**, **www.linuxworld.org**, and **www.linux.org**. Some specialize in particular areas, such as **linuxheadquarters.org** for guides on Linux software and **www.linuxgames.com** for the latest games ported for Linux. Currently, many Linux Web sites provide news, information, and articles on Linux developments, as well as documentation, software links, and other resources. These are listed in Table 1-7.

Table 1-7: Linux Information and News Sites

URL	Internet Site
www.linuxdoc.org	Web site for Linux Documentation Project
www.lwn.net	Linux Weekly News
www.linux.com	Linux.com
www.linuxtoday.com	Linux Today
www.linuxplanet.com	Linux Planet
www.linuxpower.org	Linux Power
www.linuxfocus.org	Linux Focus
www.linuxworld.org	Linux World
www.linuxmall.com	Linux Mall
www.linuxjournal.com	Linux Journal
www.linuxgazette.com	Linux Gazette
www.linux.magazine.com	Linux Magazine
www.linux.org	Linux Online
www.li.org	Linux International Web site
www.linux.org.uk	Linux European Web site

Table 1-7: Linux Information and News Sites

URL	Internet Site
linuxheadquarters.com	Linux guides and software
slashdot.org	Linux forum
webwatcher.org	Linux Web site watcher
www.opensource.org	Open source Information

Distribution FTP and Web sites, such as **www.redhat.com** and **ftp.redhat.com**, provide extensive Linux documentation and software. The **www.gnome.org** site holds software and documentation for the Gnome desktop, while **apps.kde.com** holds software and documentation for the KDE desktop. The tables in this chapter list many of the available sites. You can find other sites through resource pages that hold links to other Web sites-for example, the Linux Web site on the World Wide Web at **www.linuxdoc.org/links.html**.

Documentation

Linux documentation has also been developed over the Internet. Much of the documentation currently available for Linux can be downloaded from Internet FTP sites. A special Linux project called the Linux Documentation Project (LDP), headed by Matt Welsh, is currently developing a complete set of Linux manuals. The documentation, at its current level, is available at the LDP home site at **www.linuxdoc.org**. Linux documentations provided by the LDP are listed in Table 1-8, along with their Internet sites.

Table 1-8: Linux Documentation Project

Sites	Web Sites
www.linuxdoc.org	LDP Web site
ftp.linuxdoc.org	LDP FTP site
Guides	Document Format
Linux Installation and Getting Started Guide	DVI, PostScript, LaTeX, PDF, and HTML
Linux User's Guide	DVI, PostScript, HTML, LaTeX, and PDF
Linux System Administrator's Guide	PostScript, PDF, LaTeX, and HTML
Linux Network Administrator's Guide	DVI, PostScript, PDF, and HTML
Linux Programmer's Guide	DVI, PostScript, PDF, LaTeX, and HTML
The Linux Kernel	HTML, LaTeX, DVI, and PostScript
Linux Kernel Hacker's Guide	DVI, PostScript, and HTML
Linux HOWTOs	HTML, PostScript, SGML, and DVI
Linux FAQs	HTML, PostScript, and DVI
Linux Man Pages	Man page format

An extensive number of mirrors are maintained for the Linux Documentation Project. You can link to any of them through a variety of sources, such as the LDP home site **www.linuxdoc.org** and **www.linuxjournal.org**. The documentation includes a user's guide, an introduction, and administration guides. These are available in text, PostScript, or Web page format. Table 1-8 lists these guides. You can also find briefer explanations, in what are referred to as HOW-TO documents. HOW-TO documents are available for different subjects, such as installation, printing, and e-mail. The documents are available at Linux FTP sites, usually in the directory **/pub/Linux/doc/HOW-TO**.

Table 1-9: Usenet Newsgroups

Newsgroup	**Title**
comp.os.linux.announce	Announcements of Linux developments
comp.os.linux.devlopment.apps	For programmers developing Linux applications
comp.os.linux.devlopment.system	For programmers working on the Linux operating system
comp.os.linux.hardware	Linux hardware specifications
comp.os.linux.admin	System administration questions
comp.os.linux.misc	Special questions and issues
comp.os.linux.setup	Installation problems
comp.os.linux.answers	Answers to command problems
comp.os.linux.help	Questions and answers for particular problems
comp.os.linux.networking	Linux network questions and issues
linux.dev.*group*	There are an extensive number of development newsgroups beginning with **linux.dev**, such as **linux.dev.admin** and **linux.dev.doc**.

You can find a listing of different Linux information sites in the file **META-FAQ** located at Linux FTP sites, usually in the directory **/pub/Linux/doc**. On the same site and directory, you can also download the Linux Software Map (LSM). This is a listing of most of the software currently available for Linux.

In addition to FTP sites, Linux Usenet newsgroups are also available. Through your Internet connection, you can access Linux newsgroups to read the comments of other Linux users and to post messages of your own. Several Linux newsgroups exist, each beginning with **comp.os.linux**. One of particular interest to the beginner is **comp.os.linux**.**help**, where you can post questions. Table 1-9 lists some of the Usenet Linux newsgroups you can check out, particularly for posting questions.

Most of the standard Linux software and documentation currently available is already included on your Red Hat CD-ROM. HOW-TO documents are all accessible in HTML format, so you can view them easily with your Web browser. In the future, though, you may need to access Linux Internet sites directly for current information and software.

Red Hat Commercial Enhancements

Red Hat offers several commercial products and services for business and e-commerce solutions. These are bundled products where Red Hat Linux is combined with other commercial and noncommercial applications to provide solutions for business. Each is accompanied with extensive support to guarantee effective implementation and ongoing reliability.

These include the Interchange E-Commerce Platform, the Stronghold Secure server, and the Credit Card Verification Service (CCVS). The Interchange E-Commerce platform allows you to manage complex catalogs with multiple vendors. Stronghold is a secure SSL Web server based on the Apache Web server. CCVS processes online payments. In addition, the Red Hat High Availability Server allows you to combine servers into clusters, providing balanced and efficient access to key resources such as servers and shared applications.

Business suite products include full versions of IBM WebSphere Application Server, Lotus Domino R5 Application Server, and DB2 Universal Database. The combination provides businesses with collaboration, database, and Web server capabilities.

Other Linux Distributions

Although there is only one standard version of Linux, there are actually several different releases. Different companies and groups have packaged Linux and Linux software in slightly different ways. Each company or group then releases the Linux package, usually on a CD-ROM. Later releases may include updated versions of programs or new software. Some of the more popular releases, aside from Red Hat, are OpenLinux, SuSE, and Debian. The Linux kernel is, of course, centrally distributed through **www.kernel.org**. All distributions use this same kernel, although it may be configured differently.

Table 1-10 lists the Web sites for several of the more popular Linux distributions. Listed here also are Linux kernel sites where the newest releases of the official Linux kernel are provided. These sites have corresponding FTP sites where you can download updates and new releases, as well as third-party software packaged for these distributions (see Table 1-11). For those not listed, check their Web sites for FTP locations.

Table 1-10: Linux Distributions and Kernel Sites

URL	**Internet Site**
www.redhat.com	Red Hat Linux
www.caldera.com	OpenLinux (Caldera)
www.suse.com	SuSE Linux
www.debian.org	Debian Linux
www.infomagic.com	Infomagic
www.linuxppc.com	LinuxPPC (Mac PowerPC version)
www.turbolinux.com	Turbo Linux
www.slackware.com	Slackware Linux Project
www.kernel.org	The Linux Kernel

Table 1-10: Linux Distributions and Kernel Sites

URL	Internet Site
www.kernelnotes.org	Linux Kernel release information
www.linux-mandrake.com	Mandrake

Table 1-11: Linux Distribution FTP Sites

URL	Internet Site
ftp.redhat.com	Red Hat Linux and updates
ftp.redhat.com/contrib	Software packaged for Red Hat Linux
ftp.caldera.com	OpenLinux (Caldera)
ftp.suse.com	SuSE Linux
ftp.debian.org	Debian Linux
ftp.linuxppc.com	LinuxPPC (Mac PowerPC version)
ftp.turbolinux.com	Turbo Linux (Pacific Hi-Tech)

OpenLinux

Caldera OpenLinux is designed for corporate commercial use. OpenLinux Linux system and software packages include all the GNU software packages, as well as the X Window System managers, Internet servers, WordPerfect, and the K Desktop. However, it does not presently include Gnome. It is POSIX compliant, adhering to UNIX standards. Caldera distributes its OpenLinux system free of charge.

Caldera has organized its OpenLinux distribution into several different packages, each geared to different markets. These include the eDesktop package, which is designed for basic workstation operations, and the eServer package, which is designed for Linux servers. The eDesktop, included with this book, provides workstation software such as the KDE Desktop. The eServer installs server software such as the mail, FTP, and DNS servers. See the Caldera Web site at www.caldera.com for more information.

Caldera also offers a line of commercial and proprietary Linux packages. Such proprietary, licensed software packages are not freely distributable. They include such products as the Novell NetWare client. Recently Caldera merged with the Santa Cruz operation, which develops and distributes SCO UNIX.

SuSE

Originally a German language-based distribution, SuSE has become very popular throughout Europe and is currently one of the fastest growing distributions worldwide. Its current distribution includes both KDE and Gnome. Its distributions include WordPerfect, OpenOffice, and KOffice. It also bundles commercial products like AdabasD and the Linux Office Suite. Currently, it supports only Intel platforms. For more information, see www.suse.com.

Debian

Debian Linux is an entirely noncommercial project, maintained by hundreds of volunteer programmers. It does, however, incorporate support for commercial products in its distribution. Debian currently maintains software associations with Corel and Sun, among others. Currently it supports Alpha, Intel, Macintosh 68K, and Sparc platforms. For more information, see **www.debian.org**. Debian Linux features a sophisticated package management system and updating tool.

Slackware

Slackware is available from numerous Internet sites, and you can order the CD from Walnut Creek Software. It includes both Gnome and KDE. The Slackware distribution takes special care to remain as closely UNIX compliant as possible. Currently, it supports only Intel platforms. See **www.slackware.com** for more information.

LinuxPPC

The LinuxPPC distribution provides versions of Linux designed exclusively for use on PowerPC machines. The distribution will run on any PowerPC machine, including IBM, Motorola, and Apple systems (including G4 and iMac machines). It provides support for the USB on Mac systems. Its current distribution includes the Gnome desktop and the Enlightenment window manager. See **www.linuxppc.com** for more information.

TurboLinux

TurboLinux provides English, Chinese, and Japanese versions of Linux. It includes several of its own packages such as TurboPkg for automatically updating applications, the TurboDesk desktop, and the Cluster Web Server. Like Red Hat, it supports RPM packages. It is currently widely distributed in East Asia. Currently, TurboLinux supports only the Intel platform, but a PowerPC version is in development. See **www.turbolinux.com** for more information.

Mandrake

Mandrake Linux is another popular Linux distribution with many of the same features as Red Hat. It focuses on providing up-to-date enhancements and an easy-to-use installation and GUI configuration. You can learn more about Mandrake at **www.linux-mandrake.com**.

Chapter 2: Installing Red Hat Linux

Overview

This chapter describes the installation procedure for Red Hat Linux. The installation includes the Linux operating system, a great many Linux applications, and a complete set of network servers. Different Linux distributions usually have their own installation programs. The Red Hat installation program is designed to be efficient and brief, while installing as many features as possible. Certain features, such as Web server support, would ordinarily require specialized and often complex configuration operations. Red Hat automatically installs and configures many of these features.

Note Red Hat provides a detailed installation manual at its Web site. The manual consists of Web pages you can view using any browser. They include detailed figures and step-by-step descriptions. Checking this manual before you install is strongly recommended. This chapter presents all the steps in the installation process but is not as detailed as the Red Hat manual. On the Red Hat Web site at **www.redhat.com**, click Support and choose the link for your version of Red Hat, such as Red Hat Linux 7.1. This presents a list of links, including the Official Red Hat Linux Installation Guide.

Installing Linux involves several steps. First, you need to determine whether your computer meets the basic hardware requirements. These days, most Intel-based PC computers do. Red Hat supports several methods for installing Linux. You can install from a local source such as a CD-ROM or a hard disk, or from a network or Internet source. For a network or Internet source, Red Hat supports NFS, FTP, and HTTP installations. With FTP, you can install from an FTP site. With HTTP, you can install from a Web site. NFS enables you to install over a local network. For a local source, you can install from a CD-ROM or a hard disk. In addition, you can start the installation process by booting from your CD-ROM, from a DOS system, or from boot disks that can then use the CD-ROM or hard disk repository. Red Hat documentation covers each of these methods in detail. This chapter deals with the installation using the CD-ROM provided by this book and a boot disk created from a boot image on the CD-ROM. This is the most common approach.

Once the installation program begins, you simply follow the instructions, screen by screen. Most of the time, you only need to make simple selections or provide yes and no answers. The installation program progresses through several phases. First, you create Linux partitions on your hard drive, and then you install the software packages. After that, you can configure your network connection, and then your X Window System for graphical user interface support. Both the X Windows System and network configurations can be performed independently at a later time.

Once your system is installed, you are ready to start it and log in. Normally you will log in using a graphical login, selecting the desktop you want, and entering your user name and password. Alternatively you can log in to a simple command line interface. From the command line, you can then invoke a desktop such as Gnome that provides you with a full graphical user interface.

You have the option of installing just the operating system, the system with a standard set of applications, or all the software available on the CD-ROM. If you choose a standard installation, you can add the uninstalled software packages later. Chapter 4 and Chapter 19 describe how you can use the GnomeRPM utility or the Red Hat Package Manager to install, or even uninstall, the software packages.

Hardware, Software, Information Requirements

Before installing Linux, you must ensure that your computer meets certain minimum hardware requirements. You also need to have certain specific information ready concerning your monitor, video card, mouse, and CD-ROM drive. All the requirements are presented in detail in the following sections. Be sure to read them carefully before you begin installation.

During the installation program, you need to provide responses based on the configuration of your computer.

Hardware Requirements

Listed here are the minimum hardware requirements for installing a standard installation of the Linux system as provided by the Red Hat CD-ROM included with this book:

- A 32-bit Intel-based personal computer. At least an Intel or compatible 80386, 80486, or Pentium class microprocessor is required.
- A 3 1/2-inch floppy disk drive (if you have a bootable CD-ROM or other means of installing Linux, you don't really need a floppy drive, but for this particular installation, I know it is necessary).
- At least 64MB RAM, though 256MB is recommended.
- At least 2GB free hard disk space; 3 to 6GB or more is recommended. The size usually increases with each new release. You need at least 3GB to load and make use of all the software packages on your CD-ROM. The standard installation of basic software packages takes 2GB, plus 64 to 512MB for swap space depending on the amount of RAM memory you have. If you have less than 1GB, you can elect to perform a minimum install, installing only the Linux kernel without most of the applications. You could later install the applications you want, one at a time.
- A 3 1/2-inch, DOS-formatted, high-density (HD) floppy disk drive, to be used to create an install disk (if you are installing from a floppy).
- A CD-ROM drive.
- Two empty DOS-formatted, 3 1/2-inch, high-density (HD) floppy disks (for installation from a floppy).

If you plan to use the X Windows graphical user interface, you will also need

- A video graphics card
- A mouse or other pointing device

Software Requirements

Only a few software requirements exist. If you intend to install using the floppy disks, you need an operating system from which you can create the disks. The DOS operating system is required to enable you to prepare your installation disks. Using a DOS system, you can access the CD-ROM and issue DOS-like commands to create your installation disks. Any type of DOS will do. You can even use the same commands on OS/2. However, you do not need DOS to run Linux. Linux is a separate operating system in its own right.

If you want to have Linux share your hard disk with another operating system, like Windows, you need certain utilities to prepare the hard disk for sharing. This way one part of your hard disk could be used for Windows and another for Linux. For Windows, you need either the defrag and fips utilities or disk management software like Partition Magic 4.0. The fips utility is provided on your CD-ROM. This utility essentially frees space by reducing the size of your current extended or primary partition. Defrag and fdisk are standard DOS utilities, usually located in your **dos** directory. Defrag is used with fips to defragment your hard disk before fips partitions it. This collects all files currently on the partition into one area, leaving all the free space grouped in one large chunk. If you are installing on a new empty hard drive and you want to use part of it for Windows, you can use fdisk to set up your Windows partitions. All these tasks can also be carried out using GNU Partd and Partition Magic, mentioned earlier.

Information Requirements

Part of adapting a powerful operating system like Linux to the PC entails making the most efficient use of the computer hardware at hand. To do so, Linux requires specific information about the computer components with which it is dealing. For example, special Linux configuration files are tailored to work with special makes and models of video cards and monitors. Before installing Linux, you need to have such information on hand. The information is usually available in the manual that came with your hardware peripherals or computer.

CD-ROM, Hard Disk, and Mouse Information

For some older SCSI CD-ROM drives, you need the manufacturer's name and model.

Decide how much of your hard drive (in megabytes) you want to dedicate to your Linux system. If you are sharing with Windows, decide how much you want for Windows and how much for Linux.

Decide how much space you want for your swap partition. Your swap partition for 7.1 should be about the same as your RAM memory, but can work with as little as 64 megs. The size of the swap partition was expanded with the 2.4 kernel. Your swap partition is used by Linux as an extension of your computer's RAM.

Find the make and model of the mouse you are using. Linux supports serial, USB, PS/2, IMPS/2, and bus mice. Most mice are supported, including Microsoft, Logitech, and Mouse Systems.

Know what time zone you are in and to what time zone your hardware clock is set. This can be either Greenwich mean time (GMT) or your local time zone.

Know which kind of port your mouse is using such as PS/2, USB, or serial port. Most systems now use a PS/2 port. For a serial port mouse, you will need to know which port it is connected to: COM1, COM2, or none.

Video and Monitor Information

Although most monitors and video cards are automatically configured during installation, you might still need to provide the manufacturer's make and model in case the detection is wrong. Find out the manufacturer for your monitor and its model, such as Iiyama VisionMaster 450 or NEC E500. Do the same for your video card-for example, Matrox Millennium G400 or ATI XPERT@Play 98 (you can find a complete list of supported cards at www.xfree86.org). This should be listed on the manuals or registration information provided with your computer. For some of the most recent monitors and video cards, and some older, uncommon ones, you may need to provide certain hardware specifications. Having this information on hand, if possible, is advisable, just in case. At the end of the installation process, you are presented with lists of video cards and monitors from which to choose your own. These lists are extensive. In case your card or monitor is not on the list, however, you need to provide certain hardware information about it. If the configuration should fail, you can always do it later using an X Window System configuration utility such as Xconfigurator and XF86Setup. Of particular importance is the monitor information, including the vertical and horizontal refresh rates.

Video Card Information You should also know the following video card information, although the chipset is most likely not necessary.

- What is the make and model of your video card?
- What chipset does your video card use?
- How much memory is on your video card?

Monitor Information What is the manufacturer and model of your monitor? Linux supports an extensive list of monitors, covering almost all current ones. Your monitor will be automatically detected and selected. Should the detection be wrong, you can find it and select it from the list. If, however, your monitor is not on this list, you may need to provide the following information. Be sure this information is correct. Should you enter a horizontal or vertical refresh rate that is too high, you can seriously damage older monitors. Newer ones will just shut down. You can choose a generic profile or you can enter information for a custom profile. To do that, you need the following information:

- The horizontal refresh rate in Hz
- The vertical refresh rate in Hz

Network Configuration Information

Except for deciding your hostname, you do not have to configure your network during installation. You can put configuration off until a later time and use network configuration utilities like Linuxconf or netcfg to perform network configuration. If the information is readily available, however, the installation procedure will automatically configure your network, placing needed entries in the appropriate configuration files. If you are on a network, you must obtain most of this information from your network administrator, unless your network information is automatically provided by a DHCP server on your network. In this case, you will only need your system's hostname. During the installation process, you will be given the option of either using DHCP or entering the network information manually.

If you are setting up a network yourself, you have to determine each piece of information. If you are using a dial-up Internet service provider, you configure your network access using a PPP dial-up utility, such as kppp or Linuxconf, after you have installed the system. The installation program will prompt you to enter in these values:

- Decide on a name for your computer (this is called a *hostname*). Your computer will be identified by this name on the Internet. Do not use "localhost"; that name is reserved for special use by your system. The hostname should be a simple alphabetic word; it can include numbers but not punctuation such as periods and backslashes. A computer's name is made up of its hostname and domain name, so turtle.mytrek.com has a host name turtle and a domain name mytrek.com.
- Your domain name.

- The Internet Protocol (IP) address assigned to your machine. Every host on the Internet is assigned an IP address. This address is a set of four numbers, separated by periods, which uniquely identifies a single location on the Internet, allowing information from other locations to reach that computer.
- Your network IP address. This address is usually similar to the IP address, but with one or more zeros at the end.
- The netmask. This is usually 255.255.255.0 for class C IP addresses. If, however, you are part of a large network, check with your network administrator.
- The broadcast address for your network, if available. Usually, your broadcast address is the same as your IP address with the number 255 used for the last number.
- If you have a gateway, you need the gateway IP address for your network.
- The IP address of any name servers your network uses.
- The NIS domain and IP address if your network uses an NIS server.
- Samba server if your network is connected to a Windows network.

Upgrade Information for Currently Installed Linux Systems

If you have a version of Linux already installed and you want to upgrade it you can either overwrite your current installation, starting new, or update the current one, keeping your current configuration settings. If you are installing a new system or just overwriting the old one, you can skip this section.

If you already have installed a previous version of Red Hat Linux (kernel 2.0 and above), you may have personalized your system with different settings that you would like to keep. If you choose the Upgrade option, rather than Install, during the installation process, these settings will be kept. All your previous configuration files are saved in files with a **.rpmsave** extension. However, Upgrade only works for Red Hat versions 3.0.3 and up.

For Red Hat versions older than 3.0.3, or for other installed Linux distributions, you should save your settings first. You may want to back up these settings anyway as a precaution. These settings are held in configuration files that you can save to a floppy disk and then use on your new system, in effect retaining your original configuration (if you use **mcopy**, be sure to use the **-t** option). You may want to preserve directories and files of data, such as Web pages used for a Web site. You may also want to save copies of software packages you have downloaded. For these and for large directories, using the following **tar** operation is best.

```
tar cvMf /dev/fd0 directory-or-package
```

Make copies of the following configuration files and any other files you want to restore. You only need to copy the files you want to restore.

Files	**Description**
/etc/X11/XF86Config	X Windows configuration file
/etc/lilo.conf	Boot manager configuration file
/etc/hosts	IP addresses of connected systems
/etc/resolv.conf	Domain name server addresses
/etc/fstab	File systems mounted on your system
/etc/passwd	Names and passwords of all users on your system

Files	Description
/home/*user*	Any home directories of users with their files on your system, where *user* is the username. (For a large number of files, use **tar cfM/dev/fd0/home/user**)
.netscape	Each home directory has its own **.netscape** subdirectory with Netscape configuration files such as your bookmark entries
Web site pages and FTP files	You may want to save any pages used for a Web site or files on an FTP site you are running. On Red Hat versions, these are located at /var//**httpd/html** and /var/ **ftpd**.

Once you have installed your system, you can mount the floppy disk and use the information to configure the newly installed versions for your applications. In many cases you may be able to copy the saved files from the floppy to your system, overwriting those initially set up. However, new versions of applications may include changes in the format of configuration files. If the formats have changed, copying old configuration files will not work. In these cases, though, the new software versions will usually include utilities for converting old configuration files to new versions. Be sure to check software documentation before you replace any configuration files. This is particularly true for Internet server configuration files. For example, to convert from inetd to xinetd configuration files, you use the inetdconvert program.

If you want to restore the **/etc/lilo.conf** file from your previous system, you must also install it, using the following command:

```
# lilo /etc/lilo.conf
```

To restore archives that you saved on multiple disks using the **tar** operation, place the first disk in the floppy drive and use the following command:

```
tar xvMf /dev/fd0
```

Opening Disk Space for Linux Partitions for Shared Hard Disks

If you are using an entire hard drive for your Linux system or if you are upgrading a currently installed Linux system and you want to use the same partitions, you can skip this section and go on to installing Linux. If, however, your Linux system is going to share a hard drive with your Windows or DOS system, you need to organize your hard drive so that part of it is used for DOS and the remaining part is free for Linux installation. How you go about this process depends on the current state of your hard disk. If you have a new hard disk and you are going to install both Windows and Linux on it, you need to be sure to install Windows on only part of the hard drive, leaving the rest free for Linux. This means specifying a size smaller than the entire hard disk for your Windows partition that you set up during the Windows install procedure. You could also use fdisk to create partitions manually for Windows that will take up only a part of the hard disk. If you want to install Linux on a hard disk that already has Windows installed on its entire area, however, you need to resize your primary or extended partition, leaving part of the disk free for Linux. The objective in each situation is to free space for Linux. When you install Linux, you will then partition and format that free space for use by Linux.

Several different options exist for partitioning your hard drive, depending on whether it already contains data you need to preserve. A commercial partitioning software such as Partition Magic and GNU Partd (**http://www.gnu.org/software/parted/**) can help you do this easily and safely. Red Hat also has an option whereby Linux can be installed on a current Windows partition, requiring no partitioning. In all cases you need to make sure that your hard drive has the available free space for installing your Linux system.

A hard disk is organized into partitions. The partitions are further formatted to the specifications of a given operating system. When you installed Windows, you first needed to create a primary partition for it on your hard disk. If you have only one disk on your hard drive, then you only have a primary partition. To add more partitions, you create an extended partition and then, within that, logical partitions. For example, if you have C, D, and E disks on your hard drive, your C disk is your primary partition and the D and E disks are logical partitions set up within your extended partition. You then used the DOS **format** operation to format each partition into a Windows disk, each identified by a letter. For example, you may have divided your disk into two partitions, one formatted as the C disk and the other as the D disk. Alternatively, you may have divided your hard disk into just one partition and formatted it as the C disk. To share your hard drive with Linux, you need to free some space by either deleting some of those partitions or reducing their size.

First, decide how much space you need for your Linux system. You probably need a minimum of 3GB, though more is recommended. As stated earlier, the basic set of Linux software packages takes up 1GB, whereas the entire set of software packages, including all their source code files, take several GBs. In addition, you need space for a Linux swap partition used to implement virtual memory-the same size as your RAM is recommended, though you can get by with as little as 64 megs.

Once you determine the space you need for your Linux system, you can then set about freeing that space on your hard drive. To see what options are best for you, you should first determine what your partitions are and their sizes. You can do this with the fdisk utility. To start this utility, type **fdisk** at the DOS prompt, and press ENTER.

```
C:\> fdisk
```

This brings up the menu of fdisk options. Choose Option 4 to display a list of all your current partitions and the size of each. Press ESC to leave the fdisk utility. You can use the DOS defrag and Linux fips or partd utilities to reduce the size of the partitions, creating free space from unused space on your hard drive. You should still make a backup of your important data for safety's sake. First, check if you already have enough unused space on your hard drive that can be used for Linux. If you do not, you must delete some files. When Windows creates and saves files, it places them in different sectors on your hard disk. Your files are spread out across your hard disk with a lot of empty space in between. This has the effect of fragmenting the remaining unused space into smaller sections, separated by files. The defrag utility performs a defragmentation process that moves all the files into adjoining space on the hard disk, thereby leaving all the unused space as one large continuous segment. Once you have defragmented your disk, you can use the fips utility to create free space using part or all of the unused space. fips is a version of fdisk designed to detect continuous unused space and remove it from its current Windows partition, opening unpartitioned free space that can then be used by Linux. All your Windows partitions and drives remain intact with all their data and programs. They are just smaller.

To run the defrag utility, enter the command **defrag**. This is a DOS command usually found in the **dos** or **windows** directory. You can also run it from Windows.

```
C:\> windows\defrag
```

Defrag displays a screen with colored blocks representing the different sectors on your hard disk. It carries out an optimization of your hard disk, moving all your used sectors, your data and programs, together on the hard disk. This may take a few minutes. When it is complete, you will see the used sectors arranged together on the screen. You can then exit the defrag utility.

Now you are ready to run the fips utility to free space. fips is located on your Red Hat Linux CD-ROM, also in the directory named **dosutils**. Change to your CD-ROM drive and run the fips utility. In the following example, the CD-ROM drive is drive E:

```
C:\> e:
E:\> \dosutils\fips
```

The fips utility displays a screen showing the amount of free space. Use your arrow keys to make the space smaller if you do not need all your free space for Linux. You should leave some free space for your Windows programs. Then press ENTER to free the space.

Creating the Red Hat Boot Disks

If your computer is fairly new, it most likely has the ability to boot from your CD-ROM. In this case, you can just use your CD-ROM as your installation disk, and skip this section (you may need to configure your BIOS to boot from your CD-ROM). If you cannot boot from your CD-ROM, you will have to create floppy install disks as described here.

You can install Red Hat using an install disk whose image is located on the Red Hat CD-ROM. You create the install disk using the MS-DOS program **rawrite** and an install disk image. The install disk has to be created on a computer that runs DOS. Install disk images exist for local installation (**boot.img**), installing from an network source like a Web site (**netimage.img**), and installing with PCMCIA support (**pcmcia.img**). Begin by first starting your computer and entering DOS. Then perform the following steps.

Insert the Red Hat CD-ROM into your CD-ROM drive. At your DOS prompt, change to your CD-ROM drive, using whatever the letter for that drive may be. For example, if your CD-ROM drive is the E drive, just type **e:** and press ENTER. Once you have changed to the CD-ROM drive, you then need to change to the **\images** directory. The install disk images are there, **boot.img**, **pcmia.img**, and **netboot.img**. The **rawrite** command is in the **dosutils** directory, **\dosutils\rawrite**.

To create the install disk, insert a blank floppy disk into your floppy drive. Now start the **rawrite** command. The **rawrite** command will actually write the disk image to your floppy disk. The **rawrite** command first prompts you for the name of the disk image file you want to copy. Enter the full name of the install image file (in this example, **boot.img**). The command then asks you to enter the letter of the floppy drive where you put your floppy disk. On many systems, this is the A drive.

```
E:\> cd images
```

```
E:\col\launch\floppy > e:\dosutils\rawrite
Enter source file name: boot.img
Enter destination drive (A or B) and press ENTER: a
```

Press ENTER to confirm that you have a blank floppy disk in the drive. **rawrite** will then copy the image file to your floppy disk, creating your install disk. When it finishes, remove your disk from the floppy drive. This is the disk that the installation procedure (described later) refers to as the *install diskette*. If you need to create a network boot disk, use **netimage.img** instead. For PCMCIA support, use **pcmcia.img**.

Installing Linux

Installing Linux involves several processes, beginning with creating Linux partitions, and then loading the Linux software, configuring your X Windows interface, installing the Linux Loader (LILO) that will boot your system, and creating new user accounts. The installation program is a screen-based program that takes you through all these processes, step by step, as one continuous procedure. You can use either your mouse or the keyboard to make selections. When you finish with a screen, click the Next button at the bottom to move to the next screen. If you need to move back to the previous screen, click the Back button. You can also use the TAB, the arrow keys, SPACEBAR, and ENTER to make selections. You have little to do other than make selections and choose options. Some screens provide a list of options from which you make a selection. In a few cases, you are asked for information you should already have if you followed the steps earlier in this chapter. You are now ready to begin installation. The steps for each part of the procedure are delineated in the following sections. This should not take more than an hour.

Starting the Installation Program

If you followed the instructions in the first part of the chapter, you have freed space on your hard drive, and created your install and module disks. Now you are ready to create your Linux partitions. To do this, you need to boot your computer using the install disk you made earlier. When you start your computer, the installation program will begin and, during the installation, you can create your Linux partitions.

You can start the installation using one of several methods. If your computer can boot from the CD-ROM, you can start the installation directly from the CD-ROM. Just place the Red Hat CD-ROM in the CD-ROM drive before you start your computer. After turning on your computer, the installation program will start up.

Note To boot from a CD-ROM, you may first have to change the boot sequence setting in your computer's BIOS so that the computer will try to boot first from the CD-ROM. This requires some technical ability and knowledge of how to set your motherboard's BIOS configuration.

If you have a DOS system installed on your hard drive, you can start up DOS and then use the **autoboot.bat** command in the **dosutils** directory to start the installation, as shown here. You have to execute this command from a DOS system, not the Windows DOS window. Only DOS can be running for this command to work.

```
e:\dosutils\autoboot.bat
```

If neither of these options is feasible for you, you can use the install floppy disk (see the previous section on creating a boot disk). This is perhaps the most fail-safe method of installing Linux. Insert the Linux install disk into your floppy drive and reboot your computer. Performing a cold boot is best: turn off the computer completely and then turn it on again with the install disk in the floppy drive.

The installation program will start, presenting you with an Introduction screen. After a moment, the following prompt will appear at the bottom of your screen:

```
boot:
```

Press ENTER. (If necessary, you can enter boot parameters as described in the Red Hat manual.) Configuration information will fill your screen as the installation program attempts to detect your hardware components automatically.

Your system then detects your hardware, providing any configuration specifications that may be needed. For example, if you have a IDE CD-Write drive, it will be configured automatically as a SCSI drive so that CD Writing software can use it (see Chapters 4 and 32). If you are installing from a floppy disk, it will detect your CD-ROM. If for some reason it cannot do so, your system will ask you to select yours from a list. If you still have difficulty, you may have to specify the CD-ROM at the boot prompt.

```
Boot: linux hdx=cdrom
```

Replace the *x* with one of the following letters, depending on the interface the unit is connected to, and whether it is configured as master or a slave: a-First IDE controller master, b-First IDE controller slave, c-Second IDE controller master, d-Second IDE controller slave.

As an alternative to the CD-ROM installation, you can copy the entire CD-ROM to a Windows partition (one large enough), and then install using that partition instead of the CD-ROM. You need to know the device name of the partition and the directory to which you should copy the CD-ROM files. When asked to choose the installation method, you can select hard disk. You then have to specify the partition name and the directory.

To perform multiple installations on different systems, Red Hat provides kickstart. With kickstart, an administrator can create a profile listing all the information needed for each stage of the installation process. Users can access the profile on a server and use it to automatically install Red Hat on their system. The profile can also be placed on an install floppy disk and used in floppy disk installations, providing the same kind of automatic installation. A kickstart profile is generated automatically when you install. You can edit it with the Kickstart Configurator to modify entries.

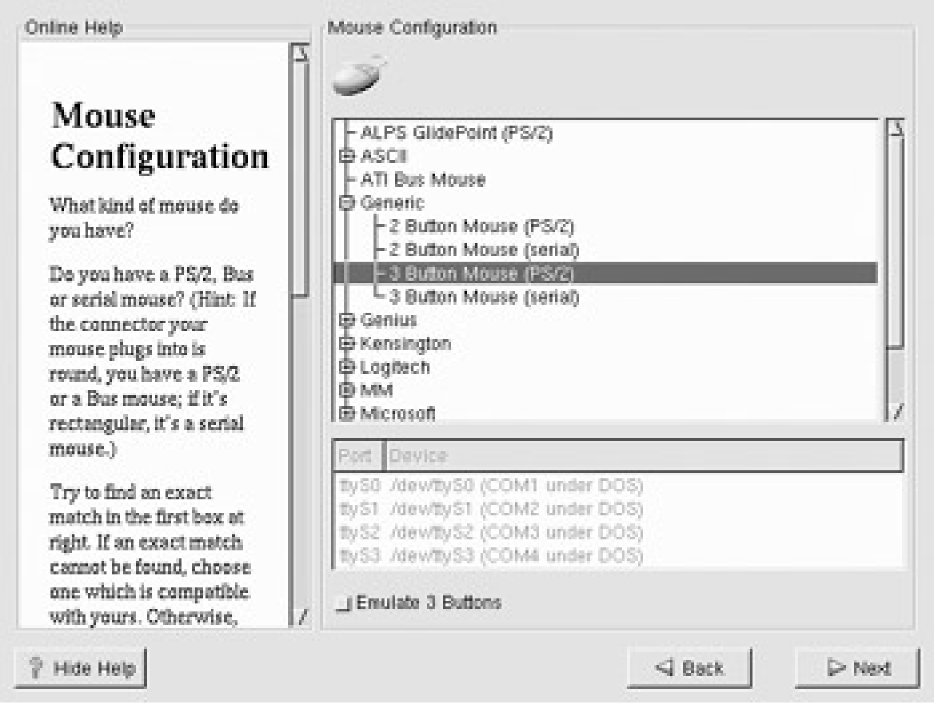
Note As each screen appears in the installation, default entries will be already selected, usually by the auto-probing capability of the installation program. Selected entries will appear highlighted. If these entries are correct, you can simply click the Next button to accept them and go on to the next screen.

Red Hat Installation

The first screen asks you to select the language you want to use. Click the language you want and then click the Next button. On the following screen, you will configure your keyboard.

The screen displays lists for selecting your keyboard model, layout, and options. A generic model works in most cases.

On the next screen, you configure your mouse (see Figure 2-1). The screen lists the different mouse brands along with specific models. One will already be automatically detected and selected for you. If the automatic detection is wrong, you can select another. Click a + symbol to expand a model list. Select your mouse. You can also check a button at the bottom of the screen to have a two-button mouse emulate a three-button mouse. A generic two-button PS/2 or serial mouse will work if your model is not listed. If you select a serial mouse, you also must select the port and device to which it is connected.



Mouse Configuration

What kind of mouse do you have?

Do you have a PS/2, Bus or serial mouse? (Hint: If the connector your mouse plugs into is round, you have a PS/2 or a Bus mouse; if it's rectangular, it's a serial mouse.)

Try to find an exact match in the first box at right. If an exact match cannot be found, choose one which is compatible with yours. Otherwise,

Port	Device
ttyS0	/dev/ttyS0 (COM1 under DOS)
ttyS1	/dev/ttyS1 (COM2 under DOS)
ttyS2	/dev/ttyS2 (COM3 under DOS)
ttyS3	/dev/ttyS3 (COM4 under DOS)

□ Emulate 3 Buttons

Figure 2-1: Mouse configuration

Now that the keyboard and mouse are configured, you can begin the installation process. The next screen displays a Welcome to Red Hat message. Click the Next button to continue.

On the following screen, labeled Install Type, you select whether you want to install a new system or upgrade a previous one. On the panel labeled Install Type, click the button for either Install or Upgrade. You use the Upgrade option to upgrade a version of Red Hat (3.0.3 or higher) that is already on your system. All your current configuration files are preserved in files with a **.rpmsave** extension. You can use them to restore your system configuration. With new software versions, configuration files sometimes change format, so be sure to check for any changes. If you do not have any complex and customized configuration files, it may be better simply to do a basic install and make what small changes you may need later.

Should you select the Install option, you can then specify a different class of installation. You can select a workstation installation, a server installation, a laptop installation, or a custom installation. Most first-time users will want to use a custom installation. The other classes of installation also perform automatic formatting of your partitions, erasing all current data on them. Use them only if you are sure of what you are doing. Workstation installations automatically erase any and all Linux partitions on your computer and use them to install Red Hat Linux.

Note Be very careful with the Server installation. The Server installation will erase all partitions on your computer hard drives, including Windows and OS/2 partitions. The Server installation is intended for computers that will operate as dedicated servers, performing only network server tasks. Should you be using Windows on the same computer that you want to use Linux on, do not select the Server option.

The workstation installations will install all the needed applications for a Linux workstation with either the Gnome or KDE desktop as your default desktop. Server installation will install server programs to enable your Linux system to operate as a network server. The custom installation enables you to choose what software packages you want installed on your system, letting you select a combination of workstation and server software or to just install everything.

Partitions

If you choose the Custom install option, an Automatic Partitioning screen is displayed with three options: Have installer automatically partition for you, manually partition with Disk Druid, and Manually partition with fdisk [for experts only]. Automatic partitioning can be used for a hard disk that will only be running Linux and have no customized requirements such as RAID devices or specialized partitions. Red Hat will detect and set up standard Linux partitions for you. Disk Druid is an easy-to-use partition manager that employs a graphical interface, while fdisk is the Linux fdisk utility that uses a simple command line interface. Use fdisk only if you are familiar with it already.

If you choose the Workstation or Server options, an Automatic Partitioning screen is displayed with three options: Manually partition with Disk Druid, Manually partition with fdisk, and Automatically partition and Remove data. If you choose either manual partition entry, you can control and select the partitions you want set up and formatted. If you choose the Remove data entry, the Workstation install will erase any current Linux partition on your system, whereas the Server install will erase all your partitions (including Windows). Should you want to back out to select a custom installation, you can just click the Back button.

If you choose a Custom installation or the Manual partition in the Automatic Partitioning screen, the Partitions screen is displayed. Here, you can manually create Linux partitions or select the one where you want to install Red Hat. The top pane displays a graphics of your partitions, and the lower pane lists the partitions in a tree format. There is a graphic for each hard drive on your system, and each partition is displayed proportionally according to the amount of space it takes up. You can edit a partition by selecting its image and clicking the Edit button. The buttons above the Partitions pane enables you to create, edit, and delete partitions. The Partitions screen is actually an interface for the Red Hat Disk Druid program, used in previous Red Hat installation programs.

You are advised to set up at least two Linux partitions: a swap partition and a root partition. The *root partition* is where the Linux system and application files are installed. If you are sharing a large hard drive with other systems like Windows, you can install the Linux root partition anywhere on the hard drive.

Except for the swap partition, when setting up a Linux partition you must specify a mountpoint. A *mountpoint* is a directory where the files on that partition are connected to the overall Linux file structure for your system. The mountpoint for your root partition is the root directory, represented by a single slash, /. The mountpoint for your boot partition is the path **/boot**. For a users' partition, it would be **/home**.

When creating a new partition, you must specify its size, though you can have the partition automatically expand to the available free space on your hard drive. The size of the swap partition should be the same size as your RAM memory, with a recommended minimum size of 64 megs. With 256 megs of RAM, you could use a 256 meg swap partition. If you have a large amount of RAM, you can make the swap partition the same size. If your disk space is limited, you should make your swap size at least 64 megs.

Be sure enough space is available for it on your hard drive. If not, you will receive an Unallocated Requested Partition message. You can free space by deleting unwanted partitions already set up or edit the new partition's entry and change its requested size. Check the entry for your hard drive in the Drive Summaries pane to find out how much free space is available on your hard drive.

To create the new partition, click the Add button to display a dialog box where you can enter the mountpoint, the size (in megabytes), the file system type, and the hard disk on which you want to create the partition. For the size, you can select a "Grow to fill disk" option to have the partition automatically expand to the size of the remaining free space on the disk. You can have this option selected for more than one partition. In that case, the partition size will be taken as a required minimum, and the remaining free space will be shared equally among the partitions. For file system type, select ext3, the Linux native type for standard Linux partitions, and select the Linux swap type for your swap partition. You can even use Disk Druid to create DOS partitions. You can also select the hard drive on which to create the partition. To make any changes later, you can edit a partition by selecting its segment in its drive graphic displayed on the upper half of the screen. Then click the Edit button.

Note With Red Hat 7.2, the standard Linux file system type is ext3, which replaces ext2.

If you want to change the size of partition that has been already created, you must first delete it and then create a new one. Remember, deleting a partition erases all data on it. To delete a partition, select it and click the Delete button.

You also have the option of creating software RAID disks. First, create partitions and select as their type Software RAID (see Chapter 32 for more details on RAID). Once you have created your partitions, you can create a RAID disk. Click the Make RAID button and then select the previously created partitions that you want to make up the RAID disk. Choose the type of RAID disk as well.

If you are formatting any old Linux partitions that still have data on them, a dialog will appear, listing them and asking you to confirm that you want to format them (new Linux partitions that you create will automatically be formatted). If you already have a Linux system and have installed Red Hat on it, you will most likely have several Linux partitions already. Some of these may be used for just the system software, such as the boot and root partitions. These should be formatted. Others may have extensive user files, such as a **/home** partition that normally holds user home directories and all the files they have created. You should *not* format such partitions.

Boot Loaders

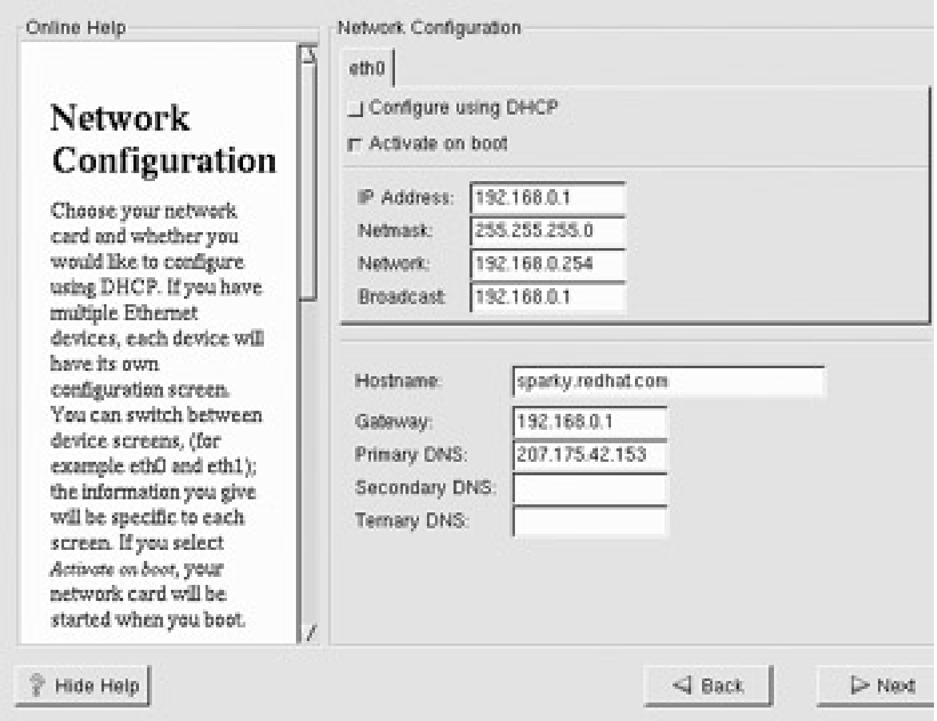
Once your partitions are prepared, you install a boot loader. You can choose either the LInux LOader (LILO) or the Grand Unified Boot loader (GRUB). GRUB is now the default boot loader. You use a boot loader to start Red Hat Linux from your hard drive. You can also use it to start any other operating system you may have installed on your computer, such as Windows. You have two choices for where to install the boot loader: the Master Boot Record (MBR) or the root partition. The recommended place is the MBR.

The Boot Loader Configuration screen lists various boot loader options. Here you can select where to install boot loader (MBR or root partition), specify a label for the Linux system (usually linux), decide whether it is to be the default system (if you have more than one operating system), and specify any kernel parameters your system may require for Linux. At the top of the screen are options for creating a boot disk selecting either the GRUB or LILO boot loaders, and *not* to install a boot loader. GRUB will be selected by default. The boot disk creation option is also automatically selected for you. You can use the boot disk to start your Linux system should there ever be a problem starting from your hard drive (for example, if you reinstall Windows on your hard drive, the boot loader is removed and you will need to use the boot disk to start Linux so that you can reinstall the boot loader).

The bottom of the screen displays a list of bootable partitions. Selecting one enables you to enter specific information for the partition in the top pane, such as the label you want to give to this partition and any kernel parameters required. The root or boot linux partition is usually given the label "linux"; a Window partition could be given a label like "win."

Network Configuration

The Network Configuration screen displays tabbed panes in the top half for the different network devices on your computer (see Figure 2-2). Click the tab for the device you want to configure. For computers already connected to a network with an Ethernet card, the tab is usually labeled eth0. Such a tab displays a pane with boxes for entering the various IP addresses for the network accessible through this device. These include the device's IP address (usually your computer's IP address), the network's address, and the broadcast address, along with the netmask. You could have a computer with several Ethernet devices, each connected to a different (or the same) network. If your network supports DHCP, you can click the DHCP button instead of manually entering in these addresses. DHCP automatically provides your computer with the needed IP addresses. You can also choose to have the device activated when your system boots or not.



Network Configuration

Choose your network card and whether you would like to configure using DHCP. If you have multiple Ethernet devices, each device will have its own configuration screen. You can switch between device screens, (for example eth0 and eth1); the information you give will be specific to each screen. If you select *Activate on boot*, your network card will be started when you boot.

□ Configure using DHCP
☑ Activate on boot

IP Address: 192.168.0.1
Netmask: 255.255.255.0
Network: 192.168.0.254
Broadcast: 192.168.0.1

Hostname: sparky.redhat.com
Gateway: 192.168.0.1
Primary DNS: 207.175.42.153
Secondary DNS:
Ternary DNS:

Figure 2-2: Network configuration

The bottom pane holds boxes for entering the IP addresses for your network's domain name servers (DNS) and gateway computer, as well as the hostname you want to give to your system. In the Hostname box, enter the fully qualified domain name for your computer.

Next, the Firewall Configuration screen lets you create basic default levels of network security. You can choose a high, medium, or low level of security. You can opt to use default firewall rules or customize your configurations specifying trusted devices and services to allow, such as Web or FTP connections. On Red Hat 7.2, the Firewall Configuration still implements older IP-Chains firewall rules. If you plan to use the newer IP-Tables, you will have to remove this configuration first (see Chapters 6 and 40).

The next screen lets you choose the language you want to use.

On the Time Zone Configuration screen, you have the option of setting the time by using a map to specify your location or by using Universal Coordinated Time (UTC) entries.

On the Account Configuration screen, you can set the root password for the root account on your system. This is the account used for system administration operations, such as installing software and managing users. On this same screen, you can also add ordinary user accounts. Click the Add button to create the account. A dialog is opened with entries for the user name, the user's full name, the password, and the password confirmation. Once you have entered in the information and clicked OK, the new user will be listed. Use the Edit button to change entries and the Delete button to remove users.

On the Authentication Configuration screen, you can add further levels of security for passwords. MD5 allows for long passwords up to 256 characters and shadow passwords that will save password information in a secure file on your system. There are tabbed panels for enabling different kinds of authentication services along with specifying their servers. There are panels for NIS, LDAP, Kerberos, and SMB. On the NIS (Network Information Service) panel, you can enable NIS and enter your NIS domain. On the LDAP panel, you can enable LDAP (Lightweight Directory Access Protocol) and specify your LDAP server if you have one; and on the Kerberos panel, you can enable Kerberos authentication. On the SMB panel, you can specify an SMB server and its workgroup for a Window network.

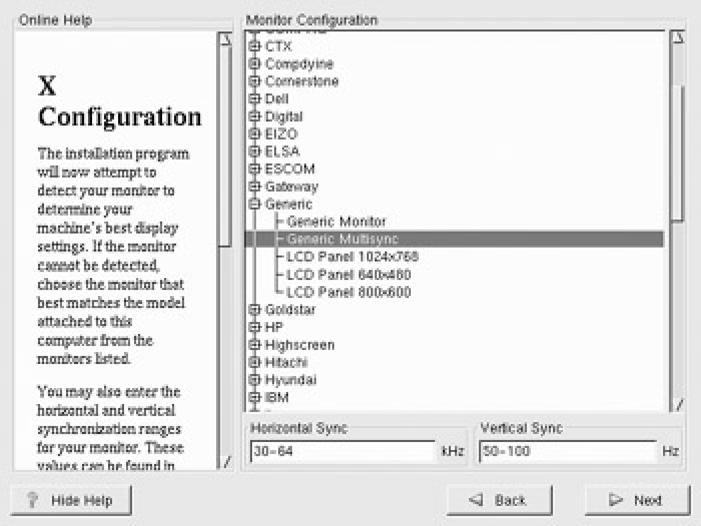
For custom installations, you are presented with a Package Group Selection screen. Here you can choose to install whole sets of packages for different categories. For example, you can install all the packages you would need for the Gnome desktop or the X Window System. To install all packages, select the Everything entry at the end of the list.

If you want to select individual software packages, click the "Select individual packages" check box. This will display a new screen with two panes, the left showing an expandable tree of software categories and the right displaying icons for the individual software packages. To select a package for installation, double-click it. You can also single-click, and then click the "Select package for installation" check box below. Whenever you single-click any icon, a description of the package is displayed on the bottom pane. Many software packages require that other software packages also be installed. This is called a *dependency*. Should you not have these already selected for installation, then an Unresolved Dependencies screen is displayed showing the packages you need to install. You can then select them for installation.

The Installing Packages screen is then displayed, which shows each package as it is installed and the progress of the installation. When the installation finishes, the Next button will become active. You can then move on to the Boot Disk Creation screen. Here, you can create a boot disk using a standard floppy disk (or you can elect to skip it).

X Window System Configuration

The X Configuration screen (see Figure 2-3) then finishes the configuration of the X Window System. This enables GUI interfaces, such as Gnome and KDE. X configuration uses the Xconfigurator utility to detect your video card and monitor automatically.



X Configuration

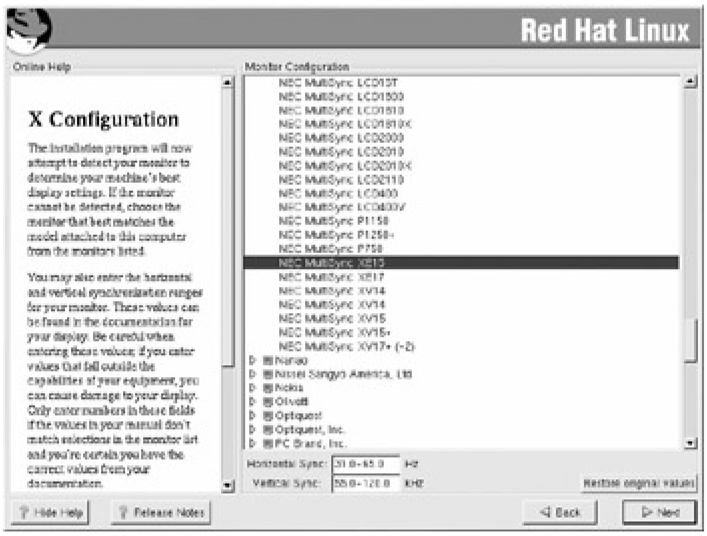
The installation program will now attempt to detect your monitor to determine your machine's best display settings. If the monitor cannot be detected, choose the monitor that best matches the model attached to this computer from the monitors listed.

You may also enter the horizontal and vertical synchronization ranges for your monitor. These values can be found in

Horizontal Sync: 30-64 kHz
Vertical Sync: 50-100 Hz

Figure 2-3: X configuration

On the Monitor Configuration screen, you select your monitor (see Figure 2-4). Initially, Xconfigurator will automatically probe your monitor and select a make and model from those listed on the screen. Check to see that the correct monitor is selected. If not, find its entry and click it. Monitor entries are organized by company in an expanding tree. If you have a Hewlett-Packard monitor, click the + symbol next to the hp entry to list all the monitors that Hewlett-Packard makes. Then click your model. When you select a monitor, the horizontal and vertical boxes at the bottom of the screen will display your monitor's horizontal sync range and vertical sync range (these values are generally available in the documentation that accompanies your monitor or from your monitor's vendor or manufacturer).



X Configuration

The installation program will now attempt to detect your monitor to determine your machine's best display settings. If the monitor cannot be detected, choose the monitor that best matches the model attached to this computer from the monitors listed.

You may also enter the horizontal and vertical synchronization ranges for your monitor. These values can be found in the documentation for your display. Be careful when entering these values; if you enter values that fall outside the capabilities of your equipment, you can cause damage to your display. Only enter numbers in these fields if the values in your manual don't match selections in the monitor list and you're certain you have the correct values from your documentation.

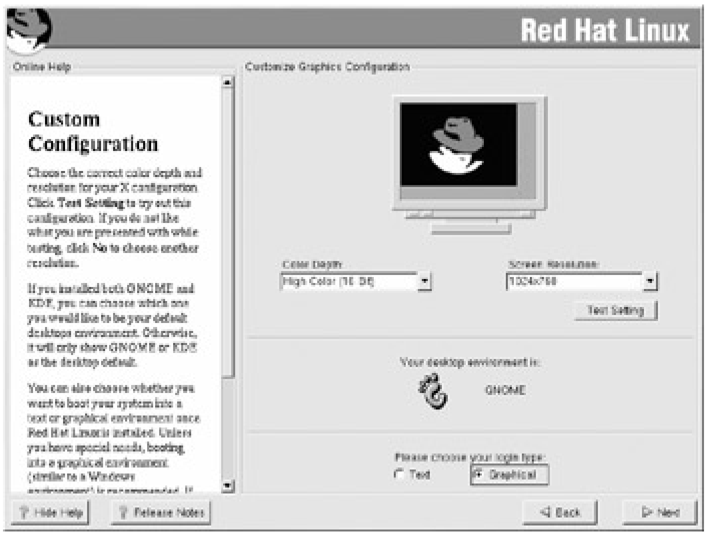
Horizontal Sync: 31.0-65.0 Hz

Vertical Sync: 55.0-120.0 KHz

Figure 2-4: Monitor configuration

If your monitor does not appear on the list, then select a Generic entry (usually Generic multisync). Xconfigurator will then supply horizontal and vertical frequency values in the labeled boxes at the bottom of the screen. Check that the correct horizontal and vertical frequencies are entered. If you enter values that are too high, you could overclock your monitor and possibly damage or destroy it. Do not select a monitor "similar" to your monitor unless you are certain the monitor you are selecting does not exceed the capabilities of your monitor.

On the Customize Graphic Configuration screen, you can set different video card configuration features such as the amount of video memory or the color depth (see Figure 2-5). If you know how much video memory is on your video card, you can check for the appropriate entry for that amount located in the drop-down menu. If an incorrect amount is selected, you can select the appropriate one.



Online Help

Custom Configuration

Choose the correct color depth and resolution for your X configuration. Click **Test Setting** to try out this configuration. If you do not like what you are presented with while testing, click No to choose another resolution.

If you installed both GNOME and KDE, you can choose which one you would like to be your default desktop environment. Otherwise, it will only show GNOME or KDE as the desktop default.

You can also choose whether you want to boot your system into a text or graphical environment once Red Hat Linux is installed. Unless you have special needs, booting into a graphical environment (similar to a Windows

Customize Graphics Configuration

Color Depth: High Color (16 Bit)

Screen Resolution: 1024x768

Your desktop environment is: GNOME

Please choose your login type: ○ Text ● Graphical

Figure 2-5: Graphic configuration

You can specify your card's resolution and color depth by selecting the specification you want from their drop-down menus. Auto-probed entries will already be selected. You can change them to other entries if you wish. An 800 x 600 resolution is usually used for 15- and 17-inch monitors, and 1024 x 768 for 19- and 21-inch monitors.

You can also choose to use a graphical login instead of the command line login. The graphical login check box will already be checked for you. If you choose this option, when you start, the Gnome Desktop Manager (GDM) will start and display a login screen where you can enter the user name and password. A Sessions submenu in the Options menu enables you to choose whether to start KDE, Gnome, or the AnotherLevel window manager. When you log in, your desktop automatically starts up. When you log out of your desktop, the GDM login window is redisplayed. Select Shutdown from the Options menu to shut down Linux.

If you do not choose this option, you will start up with the command line interface. Enter the user name at the login prompt and the password at the password prompt to log in. Use **startx** to start Gnome and **switchdesk** to switch to KDE or AnotherLevel. The **logout** command logs out, and CTRL-ALT-DEL will shut down Linux. You can also enter the **shutdown** or **halt** command while logged in to shut down Linux.

To set your default desktop, select either Gnome or KDE from the Default Desktop pop-up menu.

When you finish, Xconfigurator will generate an X Window System configuration file called **/etc/X11/XF86Config**. This is the file the X Window System uses to start up.

If you are having difficulty, you can always click the Skip X Configuration check box to skip the X Window System configuration and perform it later, after you have installed your system.

Finishing Installation

Once your boot disk is created, installation is finished. Click the Exit button of the final screen. Your system will reboot. If you chose GRUB as your boot loader, then a GRUB menu will be displayed listing Linux and other operating systems you specified such as Windows.

Use the arrow key to move to the Linux entry, if it is not already there, and press ENTER. If you chose LILO as your boot loader, then a command line boot prompt is displayed. If you set up Linux as your default operating system, just press ENTER or do nothing. Linux will then start up. If Linux is not your default, then enter the label you gave it when configuring LILO (usually "linux"). If you booted from a CD, your CD will be ejected before rebooting. If you booted directly from the CD-ROM, you may want to change your boot sequence in the BIOS back to your floppy drive.

When your system restarts, the login prompt or the GDM login screen will appear, depending upon whether you chose to have the X Window System start up automatically. You can then log into your Linux system using a login name and a password for any of the users you have set up. If you log in as the root user, you can perform administrative operations, such as installing new software or creating more users. To log in as the root user, enter **root** at the login prompt and the root user password at the password prompt.

If you are upgrading from a previously installed Red Hat Linux system and used the upgrade options, you can restore your previous configuration files, which are currently saved on your new system with the **.rpmsave** extension to their filenames. If you are upgrading from a another Linux system and have manually saved configuration files, you can restore them now. Be sure to save any current configuration files generated by your newly installed software. Check documentation for any configuration changes. You may not be able to use the old configuration files as they are.

Should you want copies of your boot disk in case you lose or damage the one you made during installation, you can create more with the **mkbootdisk** command. Enter this command in a terminal window or at the command line, and specify the kernel version number (2.4.7-10 for Red Hat 7.2), as shown here:

```
mkbootdisk 2.4.7-10
```

You can also create boot disks from the Gnome desktop using the **qmkbootdisk** tool, accessible from the Gnome system menu. This tool also lets you specify different kernels to boot (see Chapter 34). The current kernel will already be selected.

When you finish, log out of your account using the command **logout**. You then need to shut down the entire system. From the GDM login window, select Halt from the Options menu. From the command line interface, enter the **halt** command. If the system should freeze on you for any reason, you can hold down the CTRL and ALT keys and press DEL (CTRL-ALT-DEL) to safely shut it down. Never just turn it off as you do with DOS.

Should your Linux system fail to boot at any time, you can use the boot disk you created to perform an emergency boot. You can also use the install disk and, at the boot prompt, enter **boot rw root=** with the device name of the root Linux partition. For example, if your root Linux partition is **/dev/hda4**, then you would enter **boot rw root=/dev/hda4** as shown here:

```
boot> boot rw root=/dev/hda4
```

Chapter 3: Interface Basics

Overview

To start using Linux, you must know how to access your Linux system and, once you are on the system, how to execute commands and run applications. Accessing Linux involves more than just turning on your computer. Once Linux is running, you have to log into the system using a predetermined login name and password. Once on the system, you can start executing commands and running applications. You can then interact with your Linux system using either a command line interface or a graphical user interface (GUI). The Linux systems use GUI interfaces like Gnome and KDE with which you can use windows, menus, and icons to interact with your system. Most distributions, including Red Hat Linux, allow you to use a graphical login. A simple window appears with menus for selecting login options and text boxes for entering your user name and password.

Obtaining information quickly about Linux commands and utilities while logged into the system is easy. Linux has several online utilities that provide information and help. You can access an online manual that describes each command or obtain help that provides more detailed explanations of different Linux features. A complete set of manuals provided by the Linux Documentation Project is on your system and available for you to browse through or print. Both the Gnome and KDE desktops provide help systems that give you easy access to desktop, system, and application help files.

Note To make effective use of your Linux system, you must know how to configure certain features. Administrative operations such as adding users, specifying network settings, accessing CD-ROM drives, and installing software can now be performed with user-friendly system tools as well as the original command line utilities.

This chapter discusses how to access your Linux system, including logging in and out of user accounts, as well as starting the system and shutting it down. Linux commands and utilities are also covered, along with basic operations of the Gnome and KDE desktops. The chapter ends with an explanation of basic system administration operations, such as creating new user accounts and installing software packages.

User Accounts

You never directly access a Linux system. Instead, Linux sets up an interface called a *shell* through which you can interact. A Linux system can actually set up and operate several user shells at once, accommodating several users simultaneously. In fact, you can have many users working off the same computer running a Linux system. Each particular user appears to be the only one working on the system, as if Linux can set up several virtual computers and each user can then work on his or her own virtual computer. Such virtual computers are actually individually managed interfaces whereby each user interacts with the Linux system.

These user shells are frequently referred to as *accounts.* UNIX, which Linux is based on, was first used on large minicomputers and mainframes that could accommodate hundreds of users at the same time. Using one of many terminals connected to the computer, users could log into the UNIX system using their login names and passwords. All of this activity was managed by system administrators. To gain access to the system, you needed to have a user account set up for you. This was commonly known as "opening an account." A system administrator created the account on the UNIX system, assigning a login name and password for it. You then used your account to log in and use the system.

Each account is identified by a login name with access protected by a password. Of course, you can access any account if you know its login name and password. On your Linux system, you can create several accounts, logging into different ones as you choose. Other people can access your Linux system, making use of login names and passwords you provide for them. They have their own accounts on your system. Recall that in the previous chapter on installing Linux, you created a login name and password for yourself. These are what you use to access Linux regularly. When you created the login name and password, you were actually creating a new user account for yourself.

Note You can, in fact, create other new user accounts using special system administration tools. These tools become available to you when you log in as the root user. The root user is a special user account reserved for system administration tasks, such as creating users and installing new software. Basic system administration operations are discussed briefly in Chapter 5, but they are discussed in detail in Chapters 29-39. For now, you only need your regular login name and password.

Accessing Your Linux System

To access and use your Linux system, you must carefully follow required startup and shutdown procedures. You do not simply turn off and turn on your computer. If you have installed a boot loader, either GRUB or LILO, when you turn on or reset your computer, the boot loader first decides what operating system to load and run. GRUB will display a menu of operating systems to choose, whereas LILO will display a command-line prompt, as shown here:

```
LILO: linux
```

If, instead, you wait a moment or press the ENTER key, the boot loader loads the default operating system. (Recall that earlier you designated a default operating system.) If there is a Windows system listed, you can choose to run that instead.

You can think of your Linux operating system as operating on two different levels, one running on top of the other. The first level is when you start your Linux system, and the system loads and runs. It has control of your computer and all its peripherals. You still are not able to interact with it, however. After Linux starts, it displays a login prompt, waiting for a user to come along and log into the system to start using it. To gain access to Linux, you have to log in first.

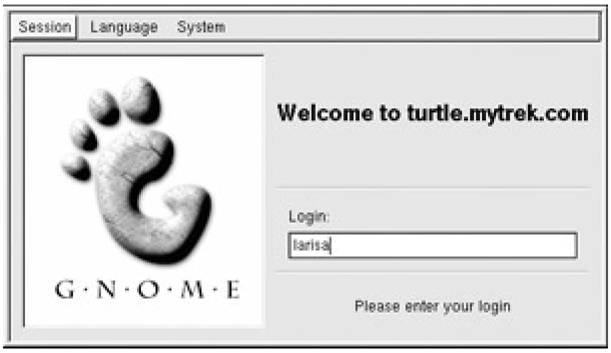
You can think of logging in and using Linux as the next level. Now you can issue commands instructing Linux to perform tasks. You can use utilities and programs such as editors or compilers, or even games. Depending on a choice you made during installation, however, you may either be interacting with the system using a simple command line interface or using the desktop directly. There are both command line login prompts and graphical login windows. In the case of most Linux distributions such as Red Hat, if you choose to use a graphical interface at the end of the installation, you are presented with a graphical login window at which you enter your login and password. If you choose not to use the graphical interface, you are presented with a simple command line prompt to enter your login name.

Gnome Display Manager: GDM

With the graphical login, your X Window System starts up immediately and displays a login window with boxes for a user login name and a password. When you enter your login name and password, and then click the OK or GO button, your default GUI starts up. On Red Hat, this is usually Gnome.

For Red Hat, graphical logins are handled by the Gnome Display Manager (GDM). The GDM manages the login interface along with authenticating a user password and login name, and then starting up a selected desktop. If problems ever occur using the X Window System display of the GUI interface, you can force a shutdown of the X Window System and the GUI with the CTRL-ALT-BACKSPACE keys. For GUI logins, it will restart the X Window System, returning you to the login screen. Also, from the GDM, you can shift to the command line interface with the CTRL-ALT-F1 keys, and then shift back to the X Window System with the CTRL-ALT-F7 keys.

When the GDM starts up, it shows a login window with a box for login, as shown in Figure 3-1. Three menus are at the top of the window, labeled Session, Language, and System. To log in, enter your login name in the Login box and press ENTER. Then you are prompted to enter your password. Do so, and then press ENTER. By default, the Gnome desktop is then started up.



Welcome to turtle.mytrek.com

Login:

larisa

Please enter your login

Figure 3-1: The Gnome Display Manager

When you log out from the desktop, you return to the GDM login window. To shut down your Linux system, click the System menu to display the entries Reboot or Halt. Select Halt to shut down your system. Alternatively, you can also shut down when you log out from Gnome. Gnome will display a logout screen with the options to log out, shut down, or reboot. Logout is the default, but selecting Shutdown will also shut down your system. Selecting reboot will shut down and restart your system. (You can also open a terminal window and enter the **shutdown**, **halt**, or **reboot** commands as described in the next section. Halt will log out and shut down your system.)

From the Session menu, you can select the desktop or window manager you want to start up. Figure 3-2 shows the default entries for Red Hat's Session menu. Here you can select KDE to start up the K Desktop instead of Gnome, among others. The Language menu lists a variety of different languages Red Hat Linux supports. Choose one to change the language interface.

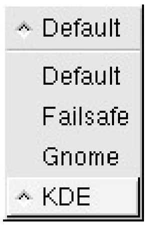


Figure 3-2: GDM Sessions menu

Command Line Interface

For the command line interface, you are initially given a login prompt. The system is now running and waiting for a user to log in and use it. You can enter your user name and password to use the system. The login prompt is preceded by the hostname you gave your system. In this example, the hostname is **turtle**. When you finish using Linux, you first log out. Linux then displays exactly the same login prompt, waiting for you or another user to log in again. This is the equivalent of the login window provided by the GDM. You can then log into another account.

```
Red hat Linux release 7.2 (Enigma)
Kernel 2.4.7-10 on i686

turtle login:
```

If you want to turn off your computer, you must first shut down Linux. If you don't shut down Linux, you could require Linux to perform a lengthy systems check when it starts up again. You shut down your system in either of two ways. First, log into an account and then enter the **halt** command. This command will log you out and shut down the system.

```
$ halt
```

Alternatively, you can use the **shutdown** command with the **-h** option. With the **-r** option, it shuts down the system and then reboots it. In the next example, the system is shut down after five minutes. To shut down the system immediately, you can use **+0** or the word **now** (see Chapter 29 for more details).

```
# shutdown -h now
```

Note Shutting down involves a series of important actions, such as unmounting file systems and shutting down any servers (never simply turn off the computer).

You can also reboot your system from the login prompt. Logging out does *not* shut down the system: it is still running and has control of your machine. To shut down and reboot your system, hold down the CTRL and ALT keys, and then press the DEL key (CTRL-ALT-DEL). You system will go through the standard shutdown procedure and then reboot your computer. At this point it is safe to turn off your computer if you wish, or just let the system restart.

When you shut down, you will see several messages as Linux shuts itself down. It is not finished until you see the "System is halted" message. If you are rebooting, Linux will shut down and then reboot your system, at which time you can turn it off if you wish. The following steps include all the startup and shutdown procedures for the command line interface:

1. Boot your computer.
2. At the boot loader, make sure the "linux" entry is selected and press ENTER. (Or, press ENTER if Linux is your default.)
3. After a few messages, the login prompt appears, and then you can log into the system and use it.
4. When you finish working in an account, you can log out. The login prompt then reappears, and you can log into another account.
5. If you are finished working on Linux and want to shut it down, enter the **halt** command while still logged into your account (**shutdown -r** will reboot).
6. At the login prompt, you can also shut down and reboot the system with CTRL-ALT-DEL. The system first shuts down and then restarts, at which time you can turn off your computer or just let the system start up again.

Once you log into an account, you can enter and execute commands. Logging into your Linux account involves two steps: entering your user name, and then your password. You already know what the login prompt looks like. Type in the login name for your user account. If you make a mistake, you can erase characters with the BACKSPACE key. In the next example, the user enters the user name **richlp** and is then prompted to enter the password:

```
Red hat Linux release 7.2 (Enigma)
Kernel 2.4.7-10 on i686

turtle login: richlp
Password:
```

When you type in your password, it does not appear on the screen. This is to protect your password from being seen by others. If you enter either the login or password incorrectly, the system will respond with the error message "Login incorrect" and will ask for your login name again, starting the login process over. You can then reenter your login name and password.

Once you enter your user name and password correctly, you are logged into the system. Your command line prompt is displayed, waiting for you to enter a command. Notice the command line prompt is a dollar sign (**$**), not a sharp sign (**#)**. The **$** is the prompt for regular users, whereas the # is the prompt solely for the root user. In this version of Linux, your prompt is preceded by the hostname and the directory you are in. Both are bounded by a set of brackets.

```
[turtle /home/richlp]$
```

To end your session, issue the **logout** or **exit** commands. This returns you to the login prompt, and Linux waits for another user to log in.

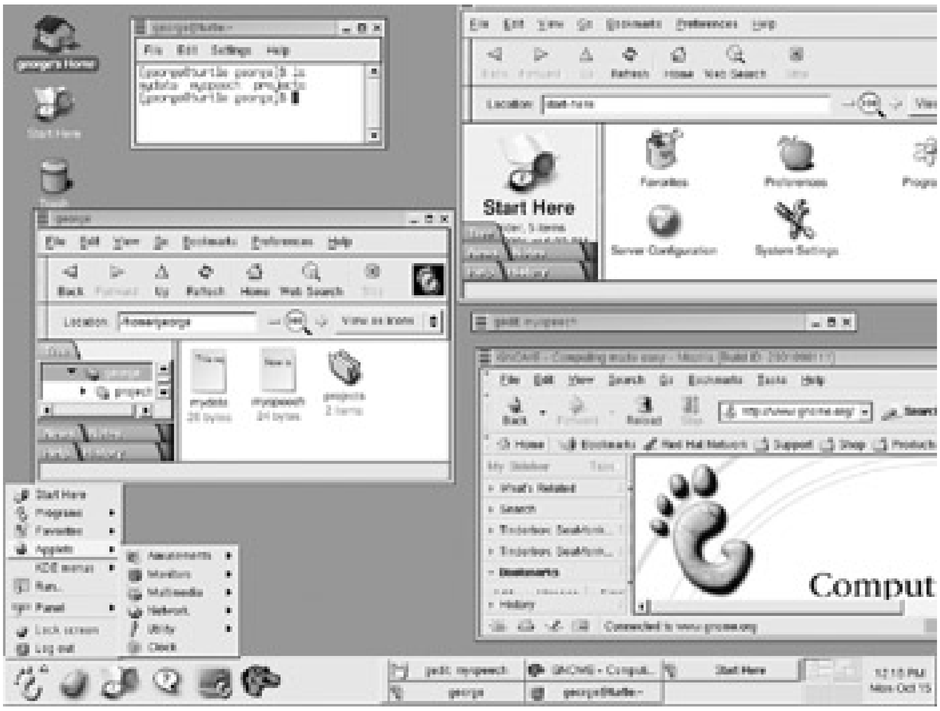
```
[turtle /home/richlp]$ logout
```

Once logged into the system, you have the option of starting an X Window System GUI, such as Gnome or KDE, and using it to interact with your Linux system. In Linux, the command **startx** starts the X Window System along with a GUI, which then enables you to interact with the system using windows, menus, and icons. On Red Hat, the **startx** command starts the Gnome desktop by default, though you can configure it to start up another desktop such as KDE or even a window manager. Once you shut down the GUI interface, you will return to your command line interface, still logged in.

On Red Hat, you can use the **switchdesk** command, while in your desktop, to switch between Gnome, KDE, or the FVWM window manager. You make your selection and then quit the desktop to return to the command line interface. When you start up the GUI again, the desktop you selected is used.

Gnome Desktop

The Gnome desktop display shown in Figure 3-3 initially displays a panel at the bottom of the screen, as well as any icons for folders and Web pages initially set up by your distribution. For Red Hat, you see several Web page icons and a folder for your home directory. The panel at the bottom of the screen contains icons for starting applications, such as Mozilla (the Mozilla logo) and the Help system (the question mark logo). You can start applications using the main menu, which you display by clicking the Gnome icon (the image of a bare footprint), located on the left side of the panel.



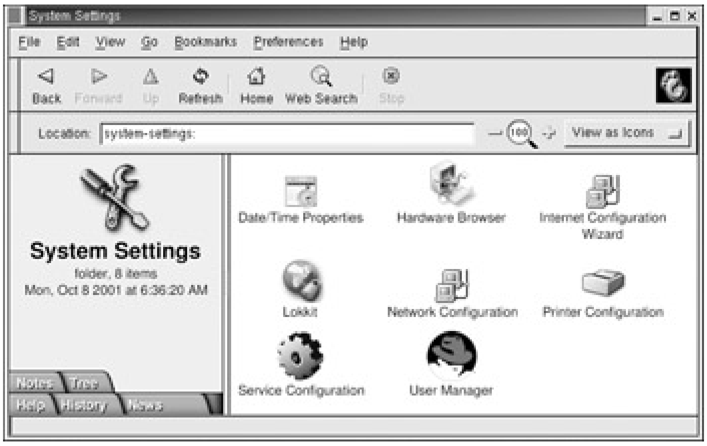
```
[george@turtle george]$ ls
mydata  myspeech  projects
[george@turtle george]$
```

Figure 3-3: Gnome

When you click the folder for your home directory on your desktop or select the File Manager entry on the main menu, a file manager window opens showing your home directory. You can display files in your home directory and use the UP ARROW button to move to the parent directory. Back and Forward buttons move through previously displayed directories. In the location window, you can enter the pathname for a directory to move directly to it. The file manager is also Internet-aware. You can use it to access remote FTP directories and to display or download their files (though it cannot display Web pages).

To move a window, left-click and drag its title bar or right-click its other borders. Each window supports Maximize, Minimize, and Close buttons, as well as a Stick Pin button. Double-clicking the title bar will "shade" a window, or rather reduce it to only its title bar; you can redisplay the window with another double-click. The desktop supports full drag-and-drop capabilities. You can drag folders, icons, and applications to the desktop or other file manager windows open to other folders. The move operation is the default drag operation. To copy files, click and drag, and then press the CTRL key before releasing the mouse button. To create links, hold down the SHIFT key when you click and drag. In most cases, you would use links for desktop icons.

The panel also contains a pager for desktop areas, which appears as four squares. Clicking a square moves you to that area. You can think of the desktop work area as being four times larger than your monitor screen, and you can use the pager to display different parts. You can configure your Gnome interface, setting features such as the background, by using the Preferences window in the Start Here window. Click the image of a compass with a map on the panel to open the Start Here window (see Figure 3-3). To configure system settings, such as adding users, installing printers, and setting up network connections, open the System Settings window in the Start Here window (see Figure 3-4). To execute a command using the command line interface, open a terminal window by clicking the image of a monitor on the panel. In that window, at the **$** prompt, type in your command.



Location: system-settings:

System Settings
folder, 8 items
Mon, Oct 8 2001 at 6:36:20 AM

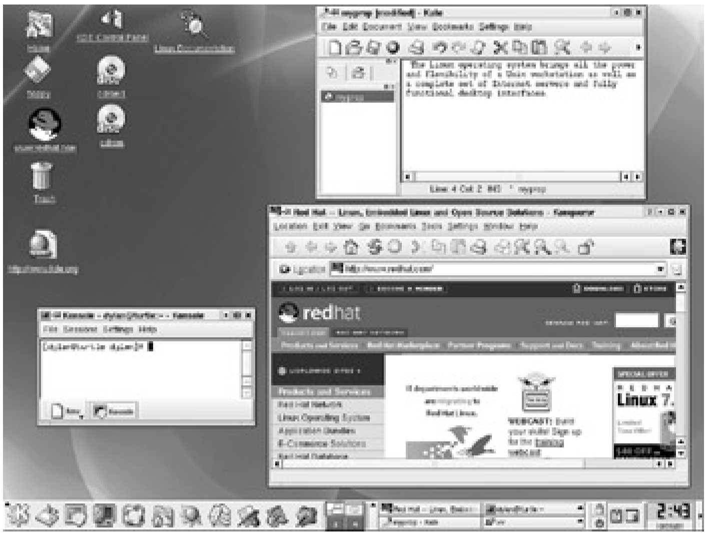
Figure 3-4: Desktop Switcher

To quit the Gnome desktop, select the logout entry at the bottom of the main menu. If you entered from a login window, you are then logged out of your account and returned to the login window. If you started Gnome from the command line, you are returned to the command line prompt, still logged into your account.

Although Gnome is the default desktop for Red Hat, you can easily switch to the KDE desktop. Red Hat installs the complete KDE desktop as part of its distribution. If you are performing a graphical login using the Gnome Desktop Manager, you can use the Session menu on the Options button to select KDE as the desktop you want to run. If you are logging into Linux using the command line interface, and then starting the desktop with the **startx** command, you can start the Gnome desktop and then use the Desktop Switcher located in the System directory to select KDE. When you quit Gnome and restart with the **startx** command, KDE is used as your desktop instead of Gnome. On KDE, you can then use the Desktop Switcher located in the Red Hat System menu to switch back to Gnome.

The K Desktop

The KDE Desktop display, shown in Figure 3-5, initially displays a panel at the bottom of the screen, as well as any icons for folders and Web pages initially set up by your distribution. In the upper-left corner, you can see a row of icons with labels like Autostart, Trash, and Printer. When a user starts KDE for the first time, the KDE Setup Wizard is run, displaying a series of four windows advising you to set up icons for KDE Web pages, as well as CD-ROM and printer icons. Initially, the KDE Wizard enables you to choose a theme, such as a Windows, KDE standard, or Mac theme. You can change this later if you want. The next windows ask if you want to add icons for your CD-ROM and printer, and links to certain Web sites, such as the KDE Web site.



The Linux operating system brings all the power and flexibility of a Unix workstation as well as a complete set of Internet servers and fully functional desktop interfaces.

Figure 3-5: The K desktop

You can start applications using the main menu, which you display by clicking the button in the panel with the large *K* on a cogwheel. This button is located on the left side of the panel. When you click the folder for your home directory on your panel (the icon of a folder with a house on it) or select the File Manager entry on the main menu, a file manager window opens, showing your home directory. You can display files in your home directory and use the UP ARROW button to move to the parent directory. Back and Forward buttons move through previously displayed directories. In the location window, you can enter the pathname for a directory to move directly to it. The file manager is also Internet-aware and a fully functional Web browser. You can use it to access remote Web and FTP sites, displaying Web pages or downloading files from an FTP site.

To move a window, click and drag its title bar or click and drag its other borders. Each window supports Stick Pin, Maximize, Minimize, and Close buttons. Double-clicking the title bar reduces the window to only its title bar (known as shading), which can redisplay with another double-click. The desktop supports full drag-and-drop capabilities. You can drag folders, icons, and applications to the desktop or to another file manager window open to other folders. Clicking the cogwheel in the right corner of a file manager window opens a duplicate window.

Selection of an icon in a file manager window is different than in other GUIs. To select an item, CTRL-click instead of making a single left-click. The single left-click is the same as a double-click on other GUIs, executing the item or opening it with its associated application. So, if you single-click on a folder icon, you open the folder (as opposed simply to selecting it). If you single-click a file, you start up the application using that file. To select items, be sure to CTRL-click them. To unselect a selected item, be sure to CTRL-click again. When you click and drag a file to the desktop or another file manager window, a pop-up menu appears, which then enables you to choose whether you want to move, copy, or create a link for the item.

The panel also contains a pager for virtual desktops. This appears as four squares. Clicking a square moves you to that desktop. You can think of the virtual desktops as separate desktops, and you can use the pager to move to the different ones. To execute a command using the command line interface, open a console window. Click the image of a monitor on the panel. In that window, at the **$** prompt, type in your command. You can modify your KDE interface at any time using the KDE Control Center. Click the image of a monitor with a circuit board on the panel or select the KDE Control Center from the main menu.

To quit the KDE desktop, select the Logout entry at the bottom of the main menu. If you entered from a login window, you are logged out of your account and returned to the login window. If you started KDE from the command line, you are returned to the command line prompt, still logged into your account.

Command Line Interface

When using the command line interface, you are given a simple prompt at which you type in your command. Even with a GUI, you sometimes need to execute commands on a command line. Linux commands make extensive use of options and arguments. Be careful to place your arguments and options in their correct order on the command line. The format for a Linux command is the command name followed by options, and then by arguments, as shown here:

```
$ command-name options arguments
```

An *option* is a one-letter code preceded by a dash, which modifies the type of action the command takes. Options and arguments may or may not be optional, depending on the command. For example, the **ls** command can take an option, **-s.** The **ls** command displays a listing of files in your directory, and the **-s** option adds the size of each file in blocks. You enter the command and its option on the command line as follows:

```
$ ls -s
```

An *argument* is data the command may need to execute its task. In many cases, this is a filename. An argument is entered as a word on the command line after any options. For example, to display the contents of a file, you can use the **more** command with the file's name as its argument. The **more** command used with the filename **mydata** would be entered on the command line as follows:

```
$ more mydata
```

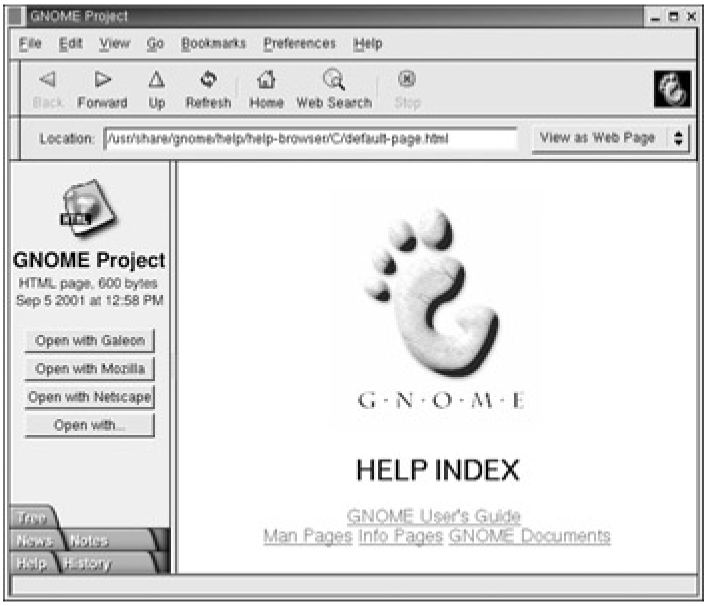
The command line is actually a buffer of text you can edit. Before you press ENTER, you can perform editing commands on the existing text. The editing capabilities provide a way to correct mistakes you may make when typing in a command and its options. The BACKSPACE and DEL keys enable you to erase the character you just typed in. With this character-erasing capability, you can BACKSPACE over the entire line if you want, erasing what you entered. CTRL-U erases the whole line and enables you to start over again at the prompt.

Note You can use the UP ARROW to redisplay your previously executed command. You can then reexecute that command, or you can edit it and execute the modified command. This is helpful when you have to repeat certain operations over and over, such as editing the same file. This is also helpful when you've already executed a command you entered incorrectly.

Help

A great deal of help is already installed on your system, as well as accessible from online sources. Both the Gnome and KDE desktops feature Help systems that use a browser-like interface to display help files. To start KDE Help, click the Book icon in the panel. Here, you can select from the KDE manual, the Linux Man pages, or the GNU info pages. KDE Help features browser capabilities, including bookmarks and history lists for documents you view.

To start the Gnome Help browser, click the icon with the question mark (?) in the panel. You can then choose from the Gnome user guide, Man pages, and info pages (see Figure 3-6). The Gnome Help browser and KDE Help Center also feature bookmarks and history lists.



Location: /usr/share/gnome/help/help-browser/C/default-page.html

Figure 3-6: Gnome Help browser

Both Gnome and KDE, along with other applications, such as Linuxconf, also provide context-sensitive help. Each KDE and Gnome application features detailed manuals that are displayed using their respective Help browsers. Also, applications like Linuxconf feature detailed context-sensitive help. Most panels on Linuxconf have Help buttons that display detailed explanations for the operations on that panel.

Note In addition, extensive help is provided online. The Red Hat desktops display Web page icons for support pages, including online manuals and tutorials.

On your system, the **/usr/share/doc** directory contains documentation files installed by each application. Within each directory, you can usually find HOW-TO documents for that application.

You can also access the online manual for Linux commands from the command line interface using the **man** command. Enter **man** with the command on which you want information.

```
$ man ls
```

Pressing either the SPACEBAR or the F key advances you to the next page. Pressing the B key moves you back a page. When you finish, press the Q key to quit the man utility and to return to the command line. You activate a search by pressing either the slash (/) or question mark (**?**). The / searches forward and the **?** searches backward. When you press the /, a line opens at the bottom of your screen, and you then enter a word to search for. Press ENTER to activate the search. You can repeat the same search by pressing the N key. You needn't reenter the pattern.

Note You can also use either the Gnome or KDE Help system to display Man pages.

Online Documentation

When you start up your browser, a default Web page lists links for documentation both on your own system and at the Red Hat Web site. To use the Red Hat Web site, you first must be connected to the Internet. However, your CD-ROM and your system contain extensive documentation showing you how to use the desktop and take you through a detailed explanation of Linux applications, including the Vi editor and shell operations. Other documentation provides detailed tutorials on different Linux topics.

The **/usr/share/doc** directory contains the online documentation for many Linux applications, including subdirectories with the names of installed Linux applications that contain documentation, such as **readme** files. You can access the complete set of HOW-TO text files in the **/usr/share/doc/HOWTO** directory. The HOW-TO series contains detailed documentation on all Linux topics from hardware installation to network configuration. In addition, **/usr/share/doc/HOWTO/HTML** holds documentation in the form of Web pages you display with a Web browser. You can use the following URL on a Web browser, such as Netscape, to view the documents:

```
file:/usr/share/doc/HOWTO/HTML
```

Online documentation for GNU applications, such as the gcc compiler and the Emacs editor, also exists. You can access this documentation by entering the command **info.** This brings up a special screen listing different GNU applications. The info interface has its own set of commands. You can learn more about it by entering **info info**. Typing **m** opens a line at the bottom of the screen where you can enter the first few letters of the application. Pressing ENTER brings up the info file on that application.

Note You can also display info documents using either the Gnome or KDE Help browser.

Part II: Basic Setup

Chapter List

Chapter 4: System Configuration

Overview

To make effective use of your Linux system, you must know how to configure certain features. Administrative operations such as adding users, accessing CD-ROM drives, and installing software can now be performed with user-friendly system tools. This chapter discusses basic system administration operations that you need to get your system up and running, as well as to perform basic maintenance such as adding new users or printers.

There are three basic system configuration tasks that you most likely will have to deal with: user management, file system access, and printer setup. You can manage users, adding new ones and removing others. File systems such as floppy disks, CD-ROMs, or other hard drives can be attached to your system at specific directories. You can also add different kinds of printers. All of these tasks you were asked to perform during installation. You can make changes or additions easily using the administration tools described in this chapter.

When logged in as the root user, you can also perform certain configuration operations from the command line. You can manually access system configuration files, editing them and making entries yourself. For example, the domain name server entries are kept in the **/etc/resolv.conf** file. You can edit this file and type in the addresses.

> Note Configuration tools are only accessible by the root user. You will first need to log in using root as your user name and providing the root password you specified during installation.

Configuration operations can be performed either from a GUI interface such as Gnome or KDE, or they can be performed using a simple shell command line at which you type in configuration commands. Red Hat also provides a set of cursor-based configuration tools, referred to as the Text Mode Setup Utility, which can be run from any shell command line. These tools cover a variety of tasks such as mouse, network, and X Windows System configuration (network configuration is covered in Chapter 5), and are shown in Table 4-1.

GUI Administration Utilities: Linuxconf and Webmin

On Red Hat, the primary administrative tools are a set of specialized GUI-based administrative tools developed and supported by Red Hat such as for network configuration and the Red Hat PPP Dialer for modem configuration. In addition, you can also use third-party GUI administrative tools such as Linuxconf and Webmin. Both provide comprehensive administration support covering tasks from users and group management to file systems and server configuration. Linuxconf was used as the primary administrative tool in Red Hat releases 6.0-7.0, and is still included with Red Hat 7.1. A full installation of Red Hat Linux 7.1 will install Linuxconf, but a standard installation will not. If you performed a standard installation, you will have to manually install Linuxconf yourself. You can also download Linuxconf or Webmin from their Web sites , as listed in Table 4-1. Commercial administration tools are also available such as Volution from Caldera.

Table 4-1: Red Hat Configuration Tools

Red Hat Administration Tool	**Description or Site**
Linuxconf	**www.solucorp.qc.ca./linuxconf**

Table 4-1: Red Hat Configuration Tools

Red Hat Administration Tool	Description or Site
Webmin	www.webmin.com
printconf	Printer configuration tool
setuptool	Text Mode Setup Utility, cursor-based configuration tool
TimeTool	Tool to set the system time and date

Linuxconf

Linuxconf provides an extensive set of configuration options, enabling you to configure features, such as user accounts and file systems, as well as your Internet servers, dial-up connections, and LILO. The version included with Red Hat does not provide support for servers. You can access the main Linuxconf interface with its entire set of configuration options or use specialized commands that display entries for a particular task, such as configuring user's accounts or entering your network settings. The specialized commands include **userconf** for user accounts, **fsconf** for file systems, and **netconf** for networks. In all cases, you need to log in as the root user.

Linuxconf supports three interfaces: an X Window System interface, a cursor-based interface, and a Web interface. The X Window System interface runs under Gnome using gnome-linuxconf to provide Gnome desktop features. You can use the cursor-based interface from a Linux command line, and you needn't be running a GUI. The interface presents a full-screen display on which you can use arrow keys, the TAB key, the SPacebar, and the ENTER key to make selections. With the Web-based interface, you use your Web browser to make selections (though this is meant for use on local networks). Use the URL for your system with a **:98** attached, as in **turtle.mytrek.com:98**.

Tip The Gnome interface for Linuxconf is installed by a separate package called the gnome-linuxconf package. Be sure this is installed to use Linuxconf on Gnome.

Webmin

Webmin is a Web page-based interface that you can run on any Web browser by accessing port 10000 at localhost **http://localhost:10000**. The initial Webmin page, shown in Figure 4-1, will have panels for different kinds of configuration tasks such as system, hardware, and servers. For basic administration tasks, click the System panel to show icons for different system administration tasks such as managing users and mounting file systems. With Webmin, you can perform all the tasks that the Red Hat tools perform.

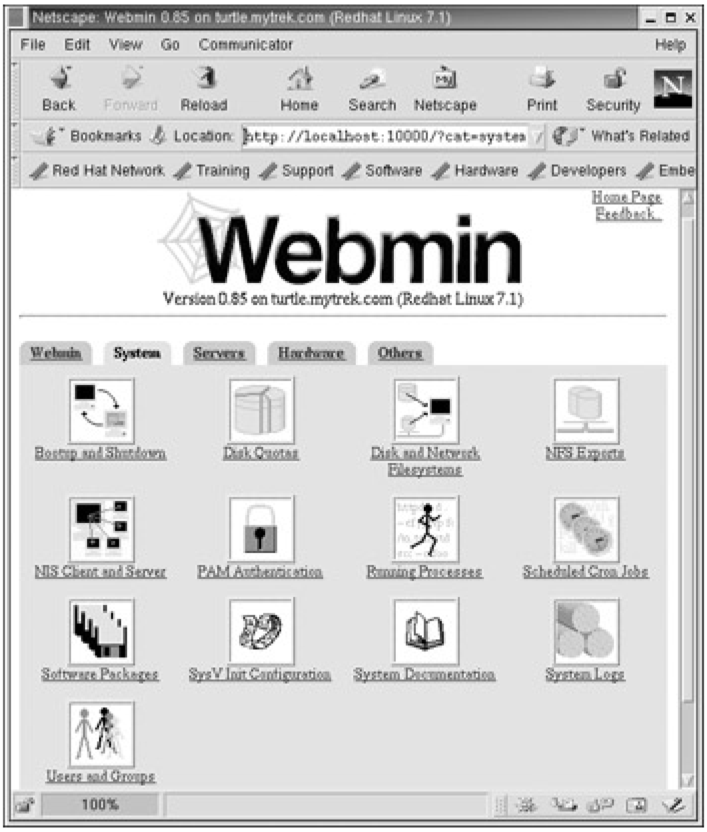
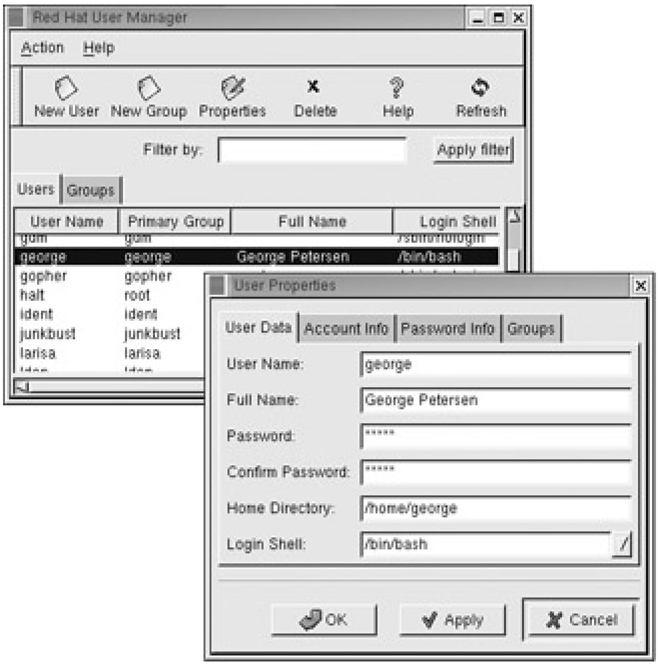


Figure 4-1: Webmin for Red Hat

For example, on Webmin you can manage users with the Users and Groups page selected from the System page. Here you can add new users, entering their user names and passwords. Current users are listed each with the user name as a link you can use to display a page for editing a user's account.

Configuring Users

Currently, the easiest and most effective way to add new users on Red Hat is to use the Red Hat User Manager. You can access it from the Gnome Desktop's Start Here window's System-Settings window. The User Manger window will display panels for listing both users and groups (see Figure 4-2). A button bar will list various tasks you can perform, including creating new users or groups, editing current ones (Properties), or deleting a selected user or group.



User Name	Primary Group	Full Name	Login Shell
gdm	gdm		/sbin/nologin
george	george	George Petersen	/bin/bash
gopher	gopher		
halt	root		
ident	ident		
junkbust	junkbust		
larisa	larisa		

User Name: george

Full Name: George Petersen

Password: *****

Confirm Password: *****

Home Directory: /home/george

Login Shell: /bin/bash

Figure 4-2: Linuxconf user account configuration

To create a new user, click on the New button. This opens a window with entries for the user name, password, login shell, along with options to create a home directory and a new group for that user. Once you have created a user, you can edit its properties to add or change features. Select the user's entry and click the Properties button. This displays a window with tabbed panels for User Data, Account Info, Password Info, and Groups. On the Groups panel, you can select the groups that the user belongs to, adding or removing group membership.

Alternatively you can use the **useradd** command to add user accounts and the **userdel** command to remove them. The following example adds the user **dylan** to the system:

```
$ useradd dylan
```

One common operation performed from the command line is to change a password. Any user can change his or her own password with the **passwd** command. The command prompts you for your current password. After entering your current password and pressing ENTER, you are then prompted for your new password. After entering the new password, you are asked to reenter it. This is to make sure you actually entered the password you intended to enter. Because password characters are not displayed when you type them, it is easy to make a mistake and to press a wrong key.

```
$ passwd
Old password:
New password:
Retype new password:
$
```

Tip From the Gnome interface you can also use the Password tool on the System menu to change your password.

Managing File Systems and CD-ROMs

Files and directories contained on different hardware devices such as floppy disks, CD-ROMs, and hard disk partitions are called *file systems.* The Linux partition you used to install your Linux system on is called the *root partition.* This is where you mounted the root file system, the root directory indicated with a single slash, /.The root partition contains the main file system with a directory tree, starting from the root and spreading out to different system and user subdirectories. To access files on another file system-say, a CD-ROM disc-you need to attach that file system to your main system. Attaching a file system is called *mounting the file system.* You first set up an empty directory to which you want to mount the file system.

Note On Red Hat, the **/mnt/cdrom** directory is already reserved for mounting CD-ROMs, and the **/mnt/floppy** directory is reserved for floppy disks. If you have more than one CD-ROM, numbered directories will be added, for example, **/mnt/cdrom1** for the second CD-ROM.

Managing CD-ROMs

The Red Hat Gnome interface also provides a simple method for mounting and unmounting a CD-ROM. Simply insert the CD-ROM into your CD-ROM drive; you then see an icon labeled CD-ROM appear on the Gnome desktop. A CD-ROM is automatically mounted. A Gnome file manager window also automatically appears, which displays the contents of the CD-ROM. You can also mount and unmount the CD-ROM using a pop-up menu on the CD-ROM icon. Right-click it to display a pop-up menu with options to mount and unmount the CD-ROM along with other options (see Figure 4-3). Selecting Unmount Volume at the bottom of the pop-up menu will automatically unmount and eject the CD-ROM from your drive. You can access the CD-ROM you placed in your CD drive by double-clicking the CD-ROM icon. Your CD-ROM drive remains locked until you select the Unmount entry that is now displayed on the pop-up menu. If you do not see an icon for your CD-ROM, you must first make it user-mountable. Use **fsconf** or Linuxconf to select the local drive and double-click the CD-ROM entry in the Local volume window. Then, on the Options panel, select the user-mountable option. Click Act/Changes to register the change. Then right-click the desktop and select Rescan Desktop Shortcuts from the pop-up menu.

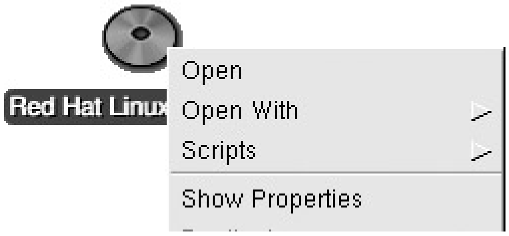


Figure 4-3: Gnome CD-ROM icon

You can also perform simple mount and unmount operations using the Disk Management tool accessible from the Gnome System menu. This tool will list all the file systems that can be mounted and will display buttons for mounting or unmounting them.

From any shell command line, you can also easily mount and unmount file systems with the **mount** and **umount** commands. To mount your CD-ROM, you only have to enter the command **mount** and the directory **/mnt/cdrom**. You can then access the contents of the CD-ROM at the **/mnt/cdrom** directory.

```
$ mount /mnt/cdrom
```

When you finish, unmount the CD-ROM with the **umount** command.

```
$ umount /mnt/cdrom
```

Note You can also manually mount and unmount floppy disks and hard disk partitions. See Chapter 32 for a detailed discussion.

Installing IDE CD-R and CD-RW Disks

If your system has a CD write (CD-R) or read/write (CD-RW) drive that uses an IDE interface, it may have been detected during installation. To support CD-R and CD-RW IDE drives, a kernel module called ide-scsi has to be loaded. The installation process will detect your CD-R or CD-RW and configure your system to automatically load the ide-scsi module. In that case your CD-R or CD-RW drive is ready to use. You can check to see if your CD-R or CD-RW drive was configured correctly by entering the following command. Information about your SCSI drives should be displayed.

```
cdrecord -scanbus
```

If configured correctly there should be an entry in your **/etc/lilo.conf** file for an append line that loads the module for your CD-R or CD-RW device. GRUB will add the argument to the command executed from its menu, which you can edit if you want.

```
append="hdc=ide-scsi"
```

Note SCSI CD-R and CD-RW drives will be automatically configured during the install process.

In this case, hdd is the device name for a CD-RW drive. There are four possible IDE devices on standard PCs, corresponding to the four primary and secondary master and slave IDE ports. The device name used in Linux depends on what IDE port you connected your CD-ROM or CD-R/CD-RW drives to. The primary master IDE port is hda and is usually used for an IDE hard drive. The other IDE ports are usually used for the CD drive. The primary slave IDE port is hdb, the secondary master is hdc (the most common connection), and the secondary slave is hdd. The above example is for a CD-R or CD-RW drive connected to the secondary master IDE port (hdc).

Many systems will have both a CD-R or CD-RW drive and a regular CD-ROM drive. If you want to copy CD-ROMs directly from the CD-ROM drive to the CD-RW drive, then you need to configure the CD-ROM drive as a SCSI drive. In the following example, there is an IDE CD-ROM drive on the secondary slave port (hdd). The **/etc/lilo.conf** append line would have to be modified to include the hdd drive.

```
append="hdc=ide-scsi hdd=ide-scsi"
```

When you restart, your CD-R and CD-RW-as well as CD-ROMs-should be installed as SCSI drives and can be used by CD write software like cdrecord and KreateCD. See the "Installing Software Packages" section later in this chapter for how to download and install KreateCD.

If the **cdrecord -scanbus** command still does not display any SCSI drives, then your CD-R or CD-RW drive was not detected and the ide-scsi module was not loaded. In this case, the entries in the **/etc/lilo.conf** file would be missing. Your CD-RW and CD-R drives are working as simple CD-ROMs, with no CD read/write capabilities. You can try to add this line to your **/etc/lilo.conf** file and then re-execute LILO with the **lilo** command (entered at the prompt in a terminal window).

You can also manually specify the ide-scsi module as a kernel parameter when your system boots up. GRUB uses this method. At the **boot:** prompt, enter the following kernel parameter.

```
boot:  linux hdc=ide-scsi
```

Enter as many CD drive entries as you need. For example, if you need to configure both the CD-ROM and the CD-RW, you could enter:

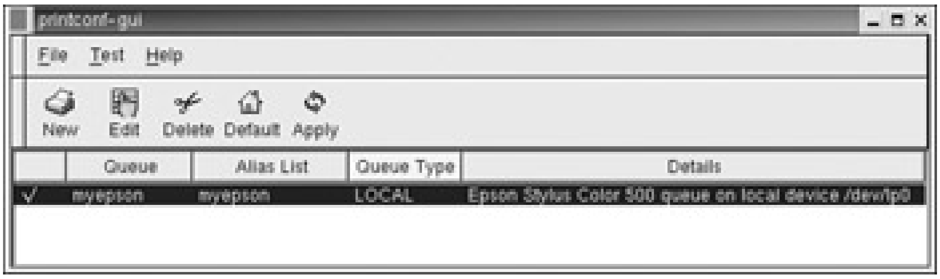
```
boot: linux hdx=ide-scsi hdd=ide-scsi
```

Your CD drives will be configured only until the system is shut down. The next time you boot up, you will have to enter the parameters again if you wish to do CD write tasks. Should this fail, you can manually install the ide-scsi module and change your CD drive device links, as described in Chapter 33.

Printer Configuration

As part of the installation procedure for Red Hat Linux, you configured a printer connected to your computer. To change configurations or to add a new printer later, you can use *printconf.* You can access printconf on the Gnome System menu. The printconf utility enables you to select the appropriate driver for your printer, as well as to set print options such as paper size and print resolutions. You can use printconf to access a printer connected directly to your local computer or to a printer on a remote system on your network (see Chapter 33).

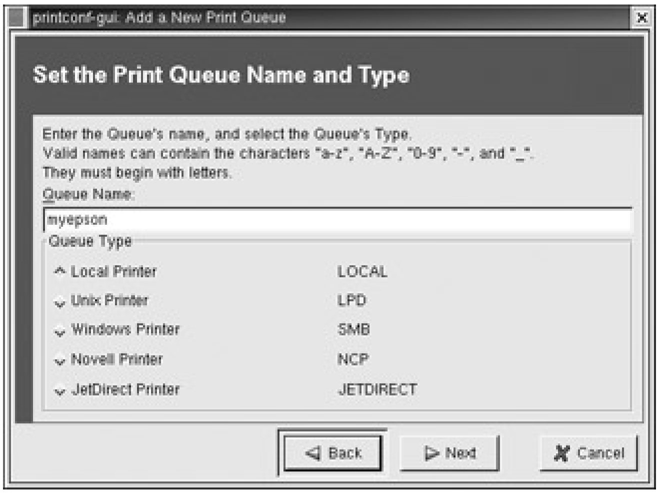
When you start up printconf, you are presented with a window that lists your installed printers (see Figure 4-4). To add a new printer, click the New button. To edit an installed printer, double-click its entry or select it and click the Edit button. Once you have made your changes, you can click on the Apply button to save your changes and restart the printer daemon. If you have more than one printer on your system, you can make one the default by selecting it and then clicking the Default button. The Delete button will remove a printer configuration.



	Queue	Alias List	Queue Type	Details
✓	myepson	myepson	LOCAL	Epson Stylus Color 500 queue on local device /dev/lp0

Figure 4-4: printconf

When you select New, a series of dialogs will take you through the process of configuring a printer, starting with entering the printer name and choosing its type (see Figure 4-5). When you edit a printer, a different dialog is displayed showing four tabbed panels: Name and Aliases, Queue Type, Driver, and Driver Options.



Set the Print Queue Name and Type

Enter the Queue's name, and select the Queue's Type.
Valid names can contain the characters "a-z", "A-Z", "0-9", "-", and "_".
They must begin with letters.

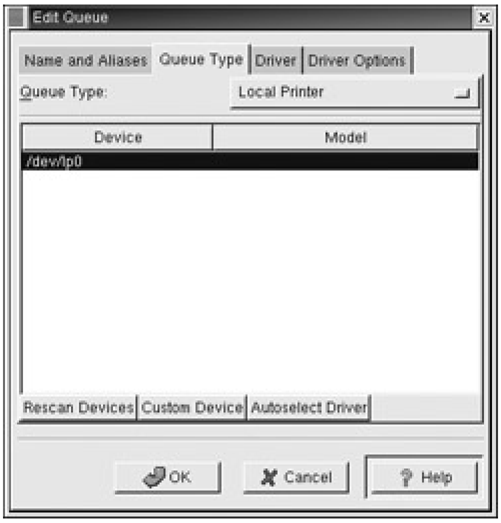
Queue Name: myepson

Queue Type

● Local Printer	LOCAL
○ Unix Printer	LPD
○ Windows Printer	SMB
○ Novell Printer	NCP
○ JetDirect Printer	JETDIRECT

Figure 4-5: printconf printer name

On the Queue panel, entries are listed for printer devices with buttons at the bottom for scanning devices, manually setting up devices, and automatically detecting a device's driver. The device is the port to which the printer is connected. For the first three parallel ports, these are **lp0**, **lp1**, **lp2;** for serial ports, these are **ttyS0**, **ttyS1**, and **ttyS2;** and so on (see Figure 4-6). From a drop-down menu, you can also specify whether the printer is local or remotely connected through a UNIX, Windows (SMB), or NetWare network.

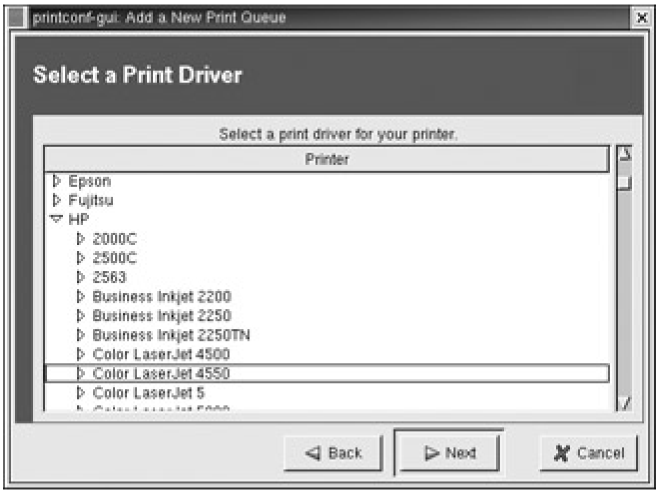


Queue Type: Local Printer

Device	Model
/dev/lp0	

Figure 4-6: printconf printer queues

For the driver selection you are presented with an expandable tree of printer types. You first select the manufacturer, such as Cannon or Apple, which then expands to a list of particular printer models (see Figure 4-7). Click yours.



printconf-gui: Add a New Print Queue

Select a Print Driver

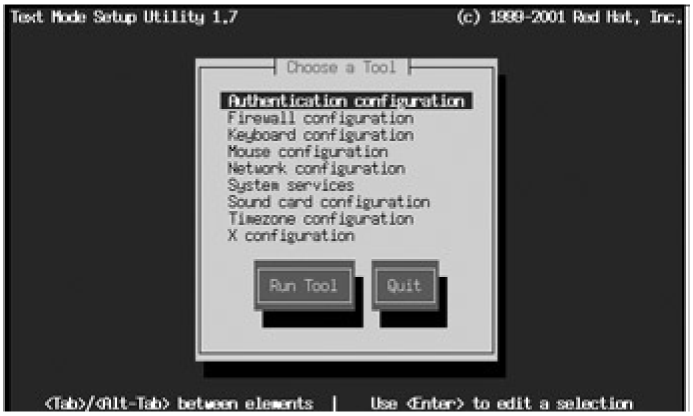
Select a print driver for your printer.

Figure 4-7: printconf printer queues

For the options selection, you can specify printer features such as paper size and resolution.

Configuration Using Red Hat Setup

Red Hat also provides a Text Mode Setup Utility (setuptool) with which you can configure different devices and system settings, such as your keyboard, mouse, and time zone. The setuptool utility is useful if you have changed any of your devices-say, installed a new mouse, keyboard, or sound card. The setuptool utility is designed to be run from the command line interface. You start the utility with the command **setup**, which you enter at a shell command line. You can also select the Text Mode Tool menu on the Gnome System menu to run setuptool from within Gnome. The setuptool utility provides a full-screen, cursor-based interface where you can use arrow, TAB, and ENTER keys to make your selections. Initially, setuptool displays a menu of configuration tools from which you can choose. Use the arrow keys to select one, and then press the TAB key to move to the Run Tool and Quit buttons. Figure 4-8 shows the initial Text Menu Setup Utility menu.



Text Mode Setup Utility 1.7 (c) 1999-2001 Red Hat, Inc.

<Tab>/<Alt-Tab> between elements | Use <Enter> to edit a selection

Figure 4-8: Red Hat Setup menu

The setuptool utility is actually an interface for running several configuration tools (see Table 4-2). You can call any of these tools separately using their commands. For example, the **kbdconfig** command starts the keyboard configuration utility that enables you to select the type of keyboard, while the **mouseconfig** command enables you to select the type of mouse.

Table 4-2: Text Mode Setup Utility

Tools	Description
setuptool	Red Hat Text Mode Setup Utility interface listing configuration tools for system and device settings
authconfig	Authentication options, such as enabling NIS, shadow passwords, and MD5 passwords
kbdconfig	Selects the keyboard type
mouseconfig	Selects the mouse type
ntsysv	Selects servers and daemons to start up at boot time
sndconfig	Detects and configures your sound card
timeconfig	Selects the time zone
Xconfigurator	Configures your X Window System for your video card and monitor

With kbdconfig, you can select the type of keyboard you are using. A text-based dialog box appears with a list of different keyboard types, which should be run from the command line interface, not a desktop.

With mouseconfig, you can select the type of mouse you are using. A cursor-based dialog box appears with a list of different mouse device types. Your system is automatically probed for the type of the mouse connected to your system, and the cursor is positioned at that entry. If you have a two-button mouse, you can select the three-button emulation option to let a simultaneous click on both the left and right mouse buttons emulate a third mouse button. This should be run from the command line interface, not from a desktop.

The ntsysv utility is a simple utility for specifying which servers and services should be automatically started at boot time (see Chapter 15). The dialog box lists the possible servers and services from which to choose. Move to the entry you want and use the SPACEBAR to toggle it on or off. An entry with an asterisk next to it is selected and is started automatically the next time you boot your system.

The sndconfig utility enables you to select and configure your sound card. It should be run from the command line interface, not from a desktop. Initially, the sndconfig utility tries to detect your sound card automatically. If the automatic detection fails, a dialog box appears with a listing of different sound cards. Select the one on your system. Another dialog box appears where you need to enter the setting for your sound card. sndconfig then tries to play sample sound and MIDI files to test the card. As an alternative to sndconfig, you can obtain, load, and configure sound drivers yourself (see Chapter 33).

Xconfigurator

One important utility is Xconfigurator, the X Window System configuration program. If you have trouble with your X Window System configuration, you can use this utility to configure it again. Xconfigurator is also helpful for updating your X Window System if you change your video card. Simply run Xconfigurator again and select the card. You can run Xconfigurator by entering the **Xconfigurator** command on the command line, or by selecting the X configuration entry in the setuptool utility's menu.

Xconfigurator first probes your system in an attempt to determine what type of video card you have. Failing that, Xconfigurator presents a list of video cards. Select your video card from the list and press ENTER. If your video card does not appear on the list, XFree86 may not support it. If you have technical knowledge about your card, however, you may choose Unlisted Card and attempt to configure it by matching your card's video chipset with one of the available X servers.

Once you select your video card, the installation program installs the appropriate XFree86 server, and Xconfigurator presents a list of monitors. If your monitor appears on the list, select it and press ENTER. If it is not on the list, select Custom. This displays a screen where you enter the horizontal sync range and vertical sync range of your monitor (these values are generally available in the documentation that accompanies your monitor or from your monitor's vendor or manufacturer). Be careful to enter the correct horizontal and vertical frequencies. If you enter values that are too high, you may overclock your monitor, which could damage older models. You should not select a monitor similar to your monitor unless you are certain the monitor you are selecting does not exceed the capabilities of your monitor.

The next screen prompts you for the amount of video memory installed on your video card. If you are not sure, please consult the documentation accompanying your video card. Choosing more memory than is available does not damage your video card, but the XFree86 server may not start correctly if you do.

If the video card you selected has a video clock chip, Xconfigurator presents a list of clock chips. The recommended choice is No Clockchip Setting because, in most cases, XFree86 can automatically detect the proper clock chip.

In the next screen, Xconfigurator prompts you to select the video modes you want to use. These are screen resolutions you may want to use. You can select one or more by moving to it and pressing the SPACEBAR. Xconfigurator then starts the X Window System and displays a dialog box asking if you can see it.

Note Xconfigurator then generates an X Window System configuration file called **/etc/X11/XF86Config**. This is the file the X Window System uses to start up.

Updating Red Hat with the Red Hat Network

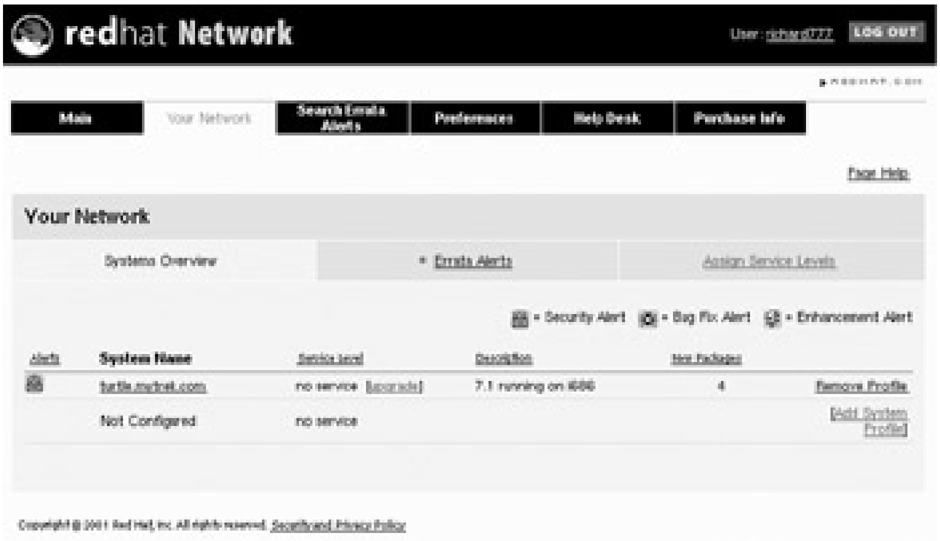
Updating your Red Hat system has become a very simple procedure, using an automatic update utility called the Red Hat Update Agent. With the Red Hat Update Agent, downloading and installing updates can be accomplished with just a few mouse clicks. The Red Hat Update Agent takes advantage of an update service provided by the Red Hat Network. Registering with and configuring access to the Red Hat Network is a very simple procedure.

New versions of distributions are often released every 6 to 12 months. In the meantime, new updates are continually being prepared for particular software packages. These are posted as updates you can download from the Red Hat FTP site and install onto your system. These include new versions of applications, servers, and even the kernel. In the period between major releases, Red Hat posts RPM package updates for software installed from your CD-ROM on its Web sites on the Red Hat Errata page at **www.redhat.com/support/errata**. Here you will find updates for different Red Hat releases. For the current release, updates are organized by security advisories, bug fixes, and package enhancement. Such updates may range from single software packages to whole components-for instance, all the core, application, and development packages issued when a new release of Gnome, KDE, or XFree86 is made available.

Red Hat provides the Red Hat Network (RHN) that you can use to update your Red Hat system securely and automatically. The Red Hat Network provides secure access to officially certified updates, including bug fixes and security advisories. You can access the RHN either through a Web browser or with the Red Hat Update Agent installed on your system. This is an improved version of the Update Agent used in 6.2. With the Red Hat Update Agent, you can automatically locate, download, and install any updates for your Red Hat system. To use the Red Hat Network, however, you first must register using the Red Hat Network Registration client. Once registered, you are then provided with a user name and password with which to access the Red Hat Network where you can set up access to the Software Manager that will automatically download your updates through the Red Hat Update client. To start the Red Hat Network, select its entry in the Gnome System menu.

The first time you use RHN, you will be asked to register. You are asked for your root user password as an added precaution. You will need to specify a user name and password, along with any other user information you want to provide. A system profile is then created, consisting of your hardware specifications and the packages you want to update. Your system is automatically probed for its hardware configuration. A list of all the RPM packages on your system is then generated with all entries selected. These will be the packages that the RHN network will update. You have the option of deselecting those you don't want to update.

You then need set up access by your system to the Software Manager, identifying your computer by its host name. This enables access to Red Hat updates. Log into the Red Hat Network using your username and password, and then click the Your Network tab at the top to display the overview screen for your network. An entry should be there for your registered system with an upgrade link under the service level. Click the upgrade link (see Figure 4-9).

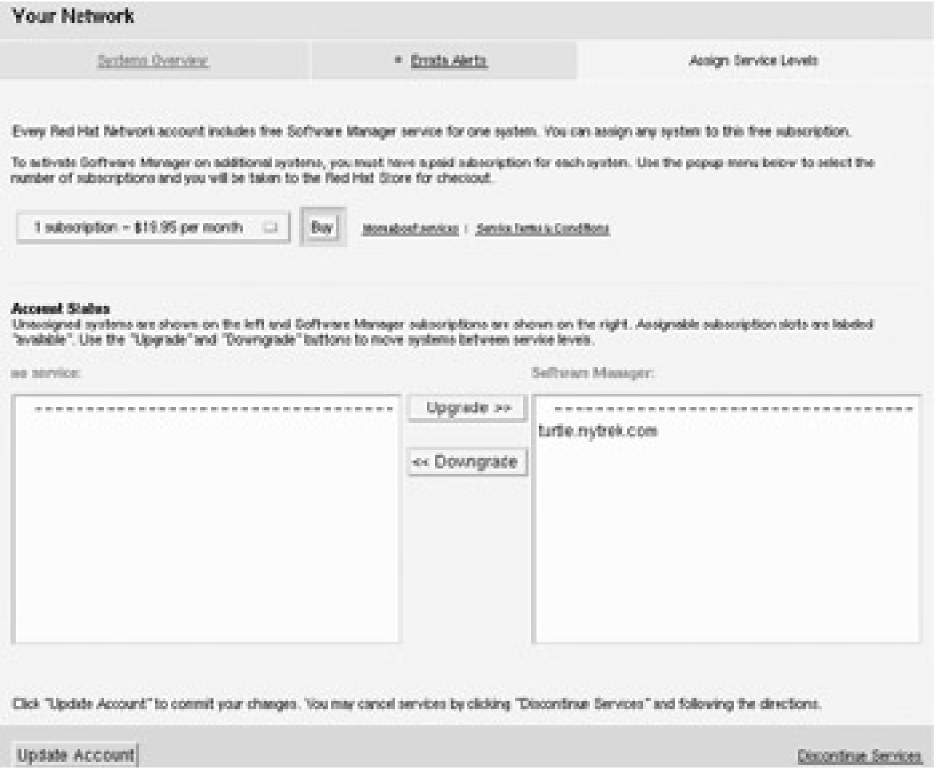


Alerts	System Name	Service Level	Description	New Packages	
▨	turtle.mytrek.com	no service [upgrade]	7.1 running on i686	4	Remove Profile
	Not Configured	no service			[Add System Profile]

Copyright © 2001 Red Hat, Inc. All rights reserved. Security and Privacy Policy

Figure 4-9: Red Hat network system overview

This displays a screen showing your status and the systems that have been added to the Software Manager. Initially your system will be in the "no service" box, shown on the left in Figure 4-10. Click its entry and click the Upgrade >> button to add it to the Software Manager box on the left. Then click the Update Account button at the bottom. Your Red Hat Update Agent now has access to the Red Hat Network and its updates. When you return to the network overview screen, you will see that your system has been activated and the upgrade link is gone.



Every Red Hat Network account includes free Software Manager service for one system. You can assign any system to this free subscription.

To activate Software Manager on additional systems, you must have a paid subscription for each system. Use the popup menu below to select the number of subscriptions and you will be taken to the Red Hat Store for checkout.

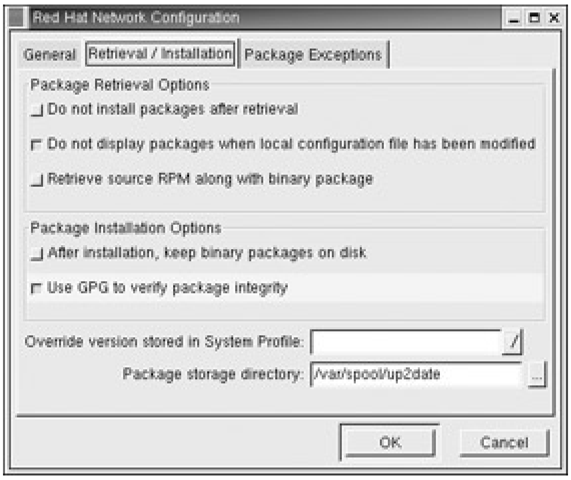
Account Status
Unassigned systems are shown on the left and Software Manager subscriptions are shown on the right. Assignable subscription slots are labeled "available". Use the "Upgrade" and "Downgrade" buttons to move systems between service levels.

turtle.mytrek.com

Click "Update Account" to commit your changes. You may cancel services by clicking "Discontinue Services" and following the directions.

Figure 4-10: Red Hat Network Software Manager access

Once you have registered and set up access to the Software Manager, you can access the RNN automatically with the Red Hat Update Agent (or manually with a Web browser). Once notified of updates, the Red Hat Update Agent will download and install them for you. To use the Red Hat Update Agent, you first have to configure it. On the Gnome desktop, select the Update Agent Configuration entry on the System menu. This displays the Configuration dialog box, shown in Figure 4-11, with three tabbed panels: General, Retrieval/Installation, and Package Exceptions. On the General panel you can enable your HTTP proxy server, should your ISP or local network use one. The Retrieval/Installation panel is where you enter download instructions and the download directory you want to use. The Package Exceptions panel holds the names of any packages you do not want to automatically update.



Package Retrieval Options

□ Do not install packages after retrieval

□ Do not display packages when local configuration file has been modified

□ Retrieve source RPM along with binary package

Package Installation Options

□ After installation, keep binary packages on disk

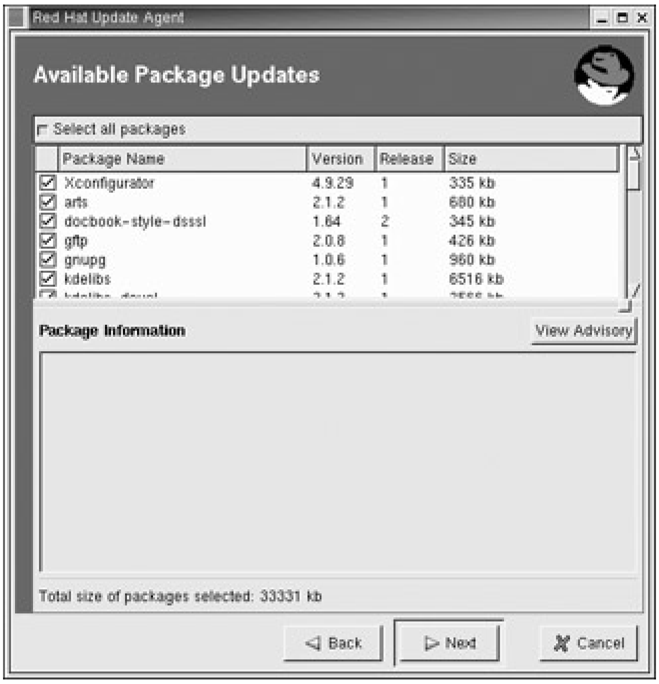
□ Use GPG to verify package integrity

Override version stored in System Profile:

Package storage directory: /var/spool/up2date

Figure 4-11: Red Hat Update Agent configuration

You are now ready to run the Red Hat Update Agent. Select Update Agent from the Gnome System menu. The first time you use the agent, you are prompted to install Red Hat GPG key. Click Yes to install. The Red Hat Update Agent then lists the possible updates it found. You can select individual packages by clicking the checkboxes nextto them or click the Select All Packages checkbox to select them all (see Figure 4-12).



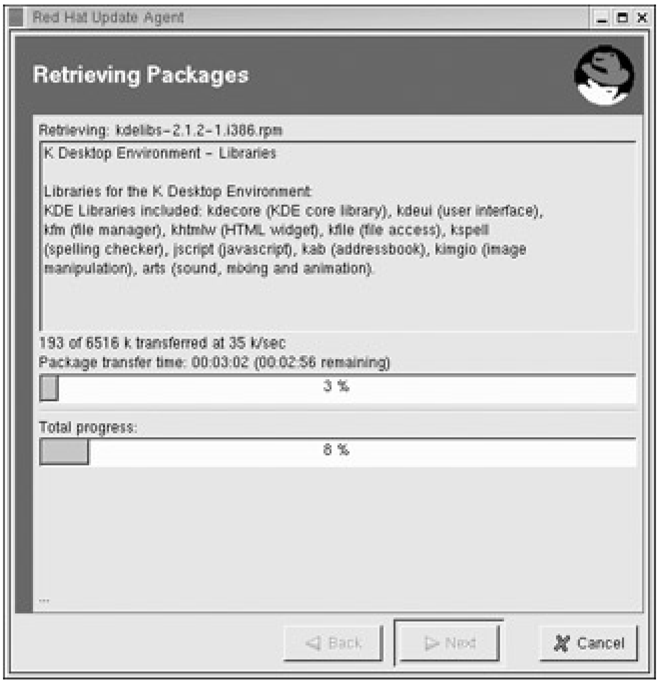
□ Select all packages

	Package Name	Version	Release	Size
☑	Xconfigurator	4.9.29	1	335 kb
☑	arts	2.1.2	1	680 kb
☑	docbook-style-dsssl	1.64	2	345 kb
☑	gftp	2.0.8	1	426 kb
☑	gnupg	1.0.6	1	960 kb
☑	kdelibs	2.1.2	1	6516 kb

Total size of packages selected: 33331 kb

Figure 4-12: Red Hat Update Agent package selection

When you click Next, the packages you selected are downloaded. Information for each package is displayed along with its download progress (See Figure 4-13). Once downloaded, the packages are installed. That is all there is to it.



Retrieving: kdelibs-2.1.2-1.i386.rpm

K Desktop Environment - Libraries

Libraries for the K Desktop Environment
KDE Libraries included: kdecore (KDE core library), kdeui (user interface), kfm (file manager), khtmlw (HTML widget), kfile (file access), kspell (spelling checker), jscript (javascript), kab (addressbook), kimgio (image manipulation), arts (sound, mixing and animation).

193 of 6516 k transferred at 35 k/sec
Package transfer time: 00:03:02 (00:02:56 remaining)

3 %

Total progress:

8 %

...

Figure 4-13: Red Hat Update Agent software download

Note Network administrators can use the Red Hat Network to download and install updates to several Red Hat systems on their network.

Tip If you installed Ximian Gnome (see Chapter 8), you can use Ximian's Red Carpet update utility to update Red Hat.

Installing Software Packages

Now that you know how to start Linux and access the root user, you can install any other software packages you may want. Installing software is an administrative function performed by the root user. Unless you chose to install all your packages during your installation, only some of the many applications and utilities available for users on Linux were installed on your system. Red Hat uses the Red Hat Package Manager (RPM) to organize Linux software into packages you can automatically install or remove. An RPM software package operates like its own installation program for a software application. A Linux software application often consists of several files that must be installed in different directories. The program itself is most likely placed in a directory called **/usr/bin**, online manual files go in another directory, and library files in yet another directory. In addition, the installation may require modification of certain configuration files on your system. The RPM software packages on your Red Hat CD-ROM perform all these tasks for you. Also, if you later decide you don't want a specific application, you can uninstall packages to remove all the files and configuration information from your system (see Chapter 19 for more details).

The RPM packages on your CD-ROMs represent only a small portion of the software packages available for Linux. You can download additional software in the form of RPM packages from distribution sites, such as **ftp.redhat.com** for Red Hat packages. In addition, these packages are organized into **lib5** and **lib6** directories: **lib5** refers to the packages using the older libraries, whereas **lib6** refers to those using the new GNU 2.*x* libraries. For Red Hat 7.0, you should use the **lib6** versions, though **lib5** versions also work. An extensive repository for RPM packages is located at **http**://rpmfind.net/. Packages here are indexed according to distribution, group, and name. This includes packages for every distribution, including previous versions of those distributions. You can also locate many of the newest Linux applications from **http://freshmeat.net** or **www.linuxapps.com**. Here, you can link to the original development sites for these applications and download documentation and the recent versions. Table 4-3 lists several Linux software sites.

Table 4-3: Linux Software Sites

Internet Site	**Description**
www.linuxapps.com	Linux software repository
freshmeat.net	New Linux software
rpmfind.net	RPM package repository
metalab.unc.edu	Mirror site for Linux software and distributions
ftp.redhat.com	Red Hat Linux and updates
sourceforge.net	Source Forge Opensource software repository and development site
apps.kde.com	KDE software applications
www.gnome.org	Gnome software applications

Installing Packages on Red Hat

On Gnome, you can use GnomeRPM to install RPM packages. Select the GnomeRPM entry in the Gnome main menu under Systems. With GnomeRPM, you can locate packages on your file system. Choose the packages you want to install. GnomeRPM lists packages already installed. You can select them to view their details and file list.

Problems can occur if the package requires that another package be installed or updated first. This is often the case where an application may need an updated version of a shared library. In this case, you will be notified of the problem and asked if you want to proceed. You can cancel at that time, and then locate and install any packages that are required first.

Updating Red Hat Manually

Red Hat also posts all updates in its **update** directory at its FTP site at **ftp.redhat.com**. You can simply download updates directly from the FTP site and install them manually. You can use an FTP client, such as ncftp, to download them all at once, including any subdirectories (see Chapter 19). Then change to that directory and use the **rpm -Uvh** command to install the packages. **U** will update a previously installed package, **v** specifies a verbose install, and **h** will display hatch symbols across the screen to show the install progress. See the following section for a more detailed explanation of RPM packages and how they work.

RPM update packages are kept in the Red Hat update site at **ftp.redhat.com** and its mirror sites. You can access the site, locate the updates for your distribution version, and then download them to your system. Then you can install them using an RPM utility, such as GnomeRPM, or the **rpm** command using the update option, **-U.** Perhaps the easiest way to do this is to use the Gnome file manager to download the files first. First, open a file manager window and enter the URL for the Red Hat update site in its location box. Then access the update site. Within that directory, move to the **current** directory, where you can see a list of all the updates. Download any new updates or all of them if this is the first update for your version. To download, open another file manager window and create a new directory to hold your update. Now open that directory. Then select and drag the update files from the file manager window for the update site to that new directory. You may not need all the files. In the case of kernel updates, you only need the kernel file for your processor: i386, i586 (Pentium), or i686 (Pentium II). As files are downloaded, a dialog box displays the filename and the percentage downloaded.

Once the update is downloaded, you can open a Terminal window and change to that new directory. You open a Terminal window by clicking the Monitor icon in the panel. Then use the **cd** command to change to that directory. If the directory name is **redup**, you would enter

```
# cd redup
```

Then issue the **rpm** command with the **Uvh** options. You should install any libraries or kernel updates first, paying attention to any dependency warnings. The ***** is a special operator that will match on filenames. For example, ***rpm** will match on all files ending with "rpm." You can use this to install selected groups of packages. The following example installs all KDE packages:

```
$ rpm -Uvh kde*rpm
```

You can also open the GnomeRPM utility, and then open the dialog box for installing packages. From the file manager window displaying the packages, you can drag and drop the files to the GnomeRPM install dialog box. You may receive error messages noting dependency requirements. You usually can safely ignore these messages. If you also receive install conflicts, you may be trying to install two versions of the same package. In that case, you must install one or the other.

Command Line Installation

If you do not have access to the desktop or you prefer to work from the command line interface, you can use the **rpm** command to manage and install software packages. The command name stands for the Red Hat Package Manager. This is the command that actually performs installation, removal, and verification of software packages. Each software package is actually an RPM package, consisting of an archive of software files and information about how to install those files. Each archive resides as a single file with a name that ends with **.rpm**, indicating it is a software package that can be installed by the Red Hat Package Manager.

You can use the **rpm** command either to install or uninstall a package. The **rpm** command uses a set of options to determine what action to take. Table 4-4 lists the set of **rpm** options. The **-i** option installs the specified software package, and the **-U** option updates a package. With an **-e** option, **rpm** uninstalls the package. A *q* placed before an *i* (**-qi**) queries the system to see if a software package is already installed and displays information about the software (**-qpi** queries an uninstalled package file). The **«-h** option provides a complete list of **rpm** options. The syntax for the **rpm** command is as follows (*rpm-package-name* is the name of the software package you want to install):

```
rpm options rpm-package-name
```

Table 4-4: rpm Options

Option	Action

Table 4-4: rpm Options

Option	Action
-U	Update package
-i	Install package
-e	Remove package
-qi	Display information for an installed package
-ql	Display file list for installed package
-qpi	Display information from an RPM package file (used for uninstalled packages)
-qpl	Display file list from an RPM package file (used for uninstalled packages)

The software package name is usually quite lengthy, including information about version and release date in its name. All end with **.rpm**. In the next example, the user installs the linuxconf package using the **rpm** command. Notice that the full filename is entered. To list the full name, you can use the **ls** command with the first few characters and an asterisk, **ls linuxconf***. You can also use the * to match the remainder of the name, as in **linuxconf-1.16*.rpm**. In most cases, you are installing packages with the **-U** option, update. Even if the package is not already installed, **-U** still installs it.

```
$ rpm -Uvh linuxconf-1.21r6-1.i386.rpm
```

When RPM performs an installation, it first checks for any dependent packages. These are other software packages with programs the application you are installing needs to use. If other dependent packages must be installed first, RPM cancels the installation and lists those packages. You can install those packages and then repeat the installation of the application. In a few situations, such as a major distribution update where packages may be installed out of order, installing without dependency checks is all right. For this, you use the **«-nodeps** option. This assumes all the needed packages are being installed, though.

To determine if a package is already installed, use the **-qi** option with **rpm**. The **-q** stands for query. To obtain a list of all the files the package has installed, as well as the directories it installed to, use the **-ql** option.

To query package files, add the **p** option. The **-qpi** option displays information about a package, and **-qpl** lists the files in it. The following example lists all the files in the Linuxconf package:

```
$ rpm -qpl linuxconf-1.21r6-1.i386.rpm
```

To remove a software package from your system, first use **rpm -qi** to make sure it is actually installed, and then use the **-e** option to uninstall it. As with the **-qi** option, you needn't use the full name of the installed file. You only need the name of the application. In the next example, the user removes the xtetris game from the system:

```
$ rpm -e xtetris
```

An important update you may need to perform is to update the Xfree86 packages. If you install a new video card or a monitor, and the current Xfree86 package does not support it, chances are the new one will. Simply download those packages from the distribution update sites and install them with the RPM update operation, as shown here:

```
$ rpm -Uvh --nodeps XFree86*rpm
```

Installing Source Code Applications

Many programs are available for Red Hat only in source code format. These programs are stored in a compressed archive that you need to decompress and then extract. The resulting source code can then be configured, compiled, and installed onto your system. The process has been simplified to the extent that it involves not much more than installing an RPM package. The following example shows how to extract, compile, and install the KreateCD program, a CD writer and ripper.

Note Be sure that you have installed all Red Hat development packages onto your system. Development packages contain the key components like the compiler, Gnome and KDE headers and libraries, and preprocessors. You cannot compile source code software without them.

1. First, locate the software-in this case, from **apps.kde.com**-and then download it to your system. KreateCD is downloaded in a file named **kreadcd-1.0.0.tar.gz**.
2. Then decompress and extract the file using the **tar** command with the **xvzf** options, as shown here.

   ```
   tar xvzf kreatcd-1.0.0.tar.gz
   ```

3. This will create a directory with the name of the software, in this case **kreatcd-1.0.0**. Change to this directory with the **cd** command.
4. `cd kreadcd-1.0.0`
5. Now issue the command **./configure**. This generates a compiler configuration for your particular system.

   ```
   ./configure
   ```

6. Compile the software with the **make** command.

   ```
   make
   ```

7. Finally, install the program with the **make install** command.

   ```
   make install
   ```

That is it. Most KDE and Gnome software will also place an entry for the program in the appropriate menus; for example, a KreateCD entry will be placed in the KDE Applications menu. You can then run KreateCD from the menu entry. You could also open a terminal window and enter the program's name.

Note You can change your display manager interface using the GDM Configurator accessible from the Gnome System menu. Here you can change the background image, set up automatic logins, and elect to display a user face browser to show a selectable image for each user on your system.

Chapter 5: Network Configuration

Overview

This chapter discusses the network configuration tools available for easily configuring network connections on Red Hat Linux. Network configuration differs depending on whether you are connected to a local area network (LAN) with an Ethernet card or you use a dial-up ISP connection. You had the opportunity to enter your LAN network settings during the installation process. For a dial-up ISP using a modem, you will have to configure your network connection using a PPP configuration utility such as the Red Hat PPP Dialer or KDE's Kppp. Table 5-1 lists the different Red Hat network configuration tools.

On Red Hat Linux, you can configure both LAN and PPP connections using the Red Hat Network Configuration tool (redhat=config=network). As an alternative to Network Configuration, you can use Webmin. For PPP connections, you can use RP3, the Red Hat PPP Dialer, as well as Kppp Network Configuration. The Red Hat PPP Dialer utility provides an easy-to-use interface with panels for login information, modem configuration, and dial-up connections. For DSL and ISDN, you can use asdl-config and isdn-config, accessible from the Internet on the System menu.

Table 5-1: Network Configuration Utilities

Network Configuration Utility	Description
redhat=config=network	Red Hat Network Configuration, access on System menu
RP3	Red Hat PPP Dialer, access on Internet menu
Webmin	Webmin network configuration, access with browser on localhost:10000
isdn-config	ISDN configuration, access on System menu with internet-config
adsl-config	DSL configuration, access on System menu with internet-config
adsl-setup	DSL configuration (command line interface), part of the rp-pppoe package (Roaring Penguin Point to Point Protocol on Ethernet)
Kppp	K Desktop PPP configuration and connection, access through internet-config on System menu
pppd	Point to Point Protocol daemon, enter on a command line
wvdial	PPP connection, enter on a command line and use connection script

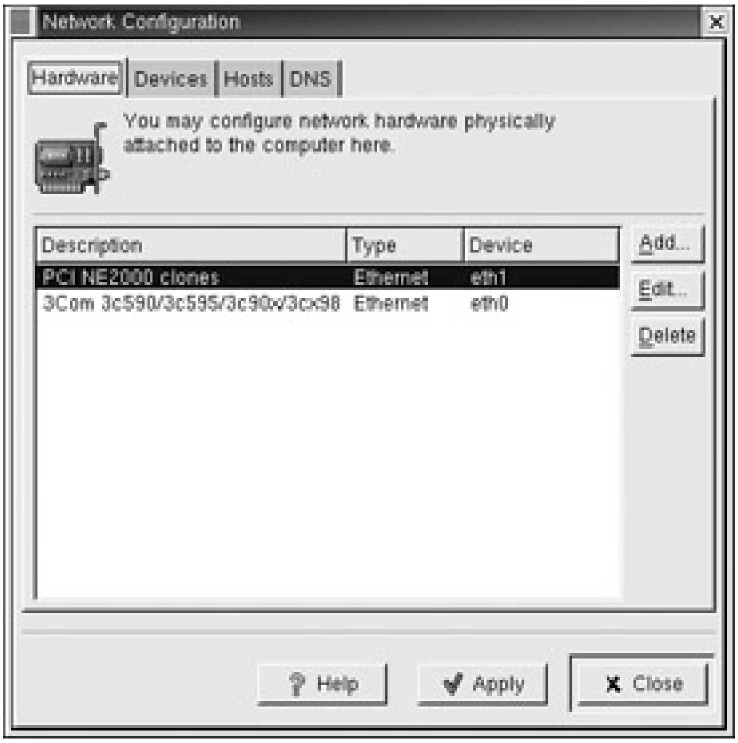
LAN

If you are on a network, you can obtain most of your network information from your network administrator or from your ISP (Internet service provider). You will need the following information. See Chapter 2 about detailed descriptions for the information you will need for your LAN configuration.

- **The device name for your network interface connection card (NIC)** This is usually an Ethernet card and has the name eth0 or eth1.
- **Hostname** Your computer will be identified by this name on the Internet. Do not use "localhost"; that name is reserved for special use by your system. The hostname should be a simple alphabetic word, which can include numbers but not punctuation such as periods and backslashes. The hostname includes the name of the host and its domain. For example, a hostname for a machine could be "turtle," whose domain is **mytrek.com**, giving it a hostname of **turtle.mytrek.com**.
- **Domain name** This is the name of your network.
- **The Internet Protocol (IP) address assigned to your machine** Every host on the Internet is assigned an IP address. This address is a set of four numbers, separated by periods, which uniquely identifies a single location on the Internet, allowing information from other locations to reach that computer.
- **Your network IP address** This address is usually similar to the IP address, but with one or more zeros at the end.
- **The netmask** This is usually 255.255.255.0 for most networks. If, however, you are part of a large network, check with your network administrator or ISP.
- **The broadcast address for your network, if available (optional)** Usually, your broadcast address is the same as your IP address with the number 255 added at the end.
- **The IP address of your network's gateway computer** This is the computer that connects your local network to a larger one like the Internet.
- **Nameservers** The IP address of the name servers your network uses. These enable the use of URLs.
- **NIS domain and IP address for an NIS server** Necessary if your network uses an NIS server (optional).

Red Hat Network Configuration

Red Hat provides an easy-to-use network configuration tool called redhat-config-network. On the Start Here System-settings window, its icon is labeled Network Configuration, and is referred here as such in this section. The Network Configuration window consists of four tabbed panels: Hardware, Devices, Hosts, and DNS (see Figure 5-1). Clicking a tab displays its panel. Basic configuration of your network requires you to specify the hostname and IP address of your own system, the IP addresses of your network's name servers and gateway, the network netmask, and your network devices. Using the Network Configuration tool, you can easily enter all this information. The DNS panel is where you enter your own system's hostname and your network's name server addresses. The Hosts panel lists host IP addresses and their domain names, including those for your own system. On the Devices panel, you add and configure your network interfaces, such as an Ethernet or PPP interface. The Hardware panel is where you list your network hardware devices. If you already configured your network during installation, your entries are already in these panels.



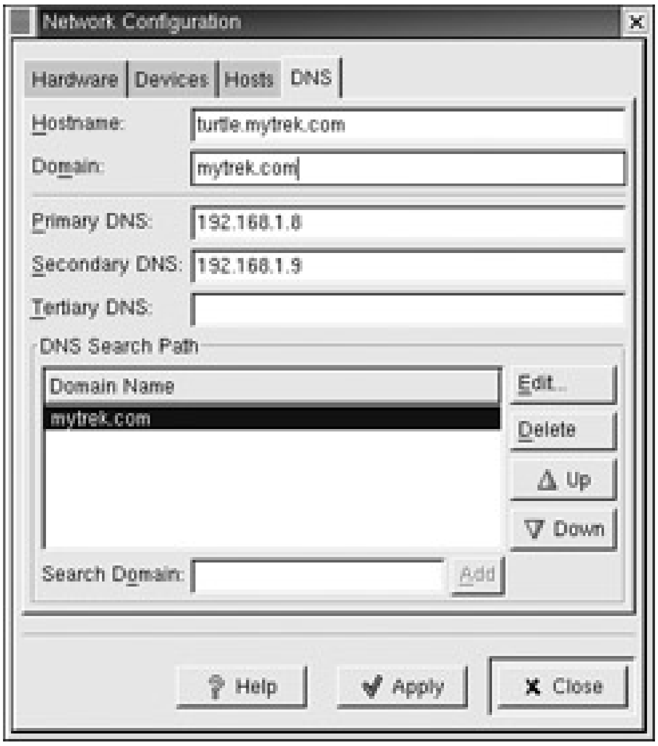
You may configure network hardware physically attached to the computer here.

Description	Type	Device
PCI NE2000 clones	Ethernet	eth1
3Com 3c590/3c595/3c90x/3cx98	Ethernet	eth0

Figure 5-1: Network Configuration Names panel

The Hardware panel list your system's network cards, such as Ethernet network interface cards (NIC), or any modems you have installed.

The DNS panel has two boxes at the top, labeled Hostname and Domain (see Figure 5-2). Here, you enter your system's fully qualified domain name and your network's domain name. For example, **turtle.mytrek.com** is the fully qualified domain name and **mytrek.com** is the domain name. There are boxes for entering the IP addresses for your system's primary, secondary, and tertiary DNS servers. You can then list search domains, with buttons for editing, deleting, or changing the priority of a domain to search. Both the search domain and the name server addresses are saved in the **/etc/resolv.conf** file. The hostname is saved to your **/etc/HOSTNAME** file.



Hostname: turtle.mytrek.com

Domain: mytrek.com

Primary DNS: 192.168.1.8

Secondary DNS: 192.168.1.9

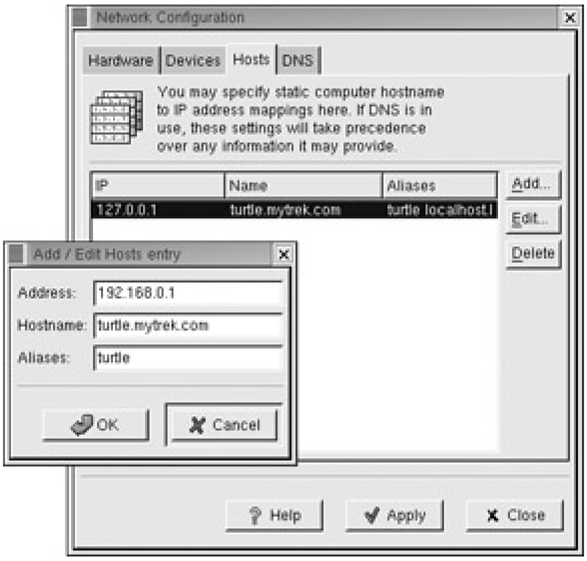
Tertiary DNS:

Domain Name
mytrek.com

Search Domain:

Figure 5-2: Network Configuration DNS panel

The Hosts panel has a single pane with Add, Edit, and Delete buttons (see Figure 5-3). This panel lists entries that associate hostnames with IP addresses. You can also add aliases (nicknames). The Hosts panel actually displays the contents of the **/etc/hosts** file and saves any entries you make to that file. To add an entry, click the Add button. A window opens with boxes for the hostname, IP address, and nicknames. When you click OK, the entry is added to the Hosts list. To edit an entry, click the Edit button and a similar window opens, enabling you to change any of the fields. To delete an entry, select it and click the Remove button.



You may specify static computer hostname to IP address mappings here. If DNS is in use, these settings will take precedence over any information it may provide.

IP	Name	Aliases
127.0.0.1	turtle.mytrek.com	turtle localhost.l

Address: 192.168.0.1

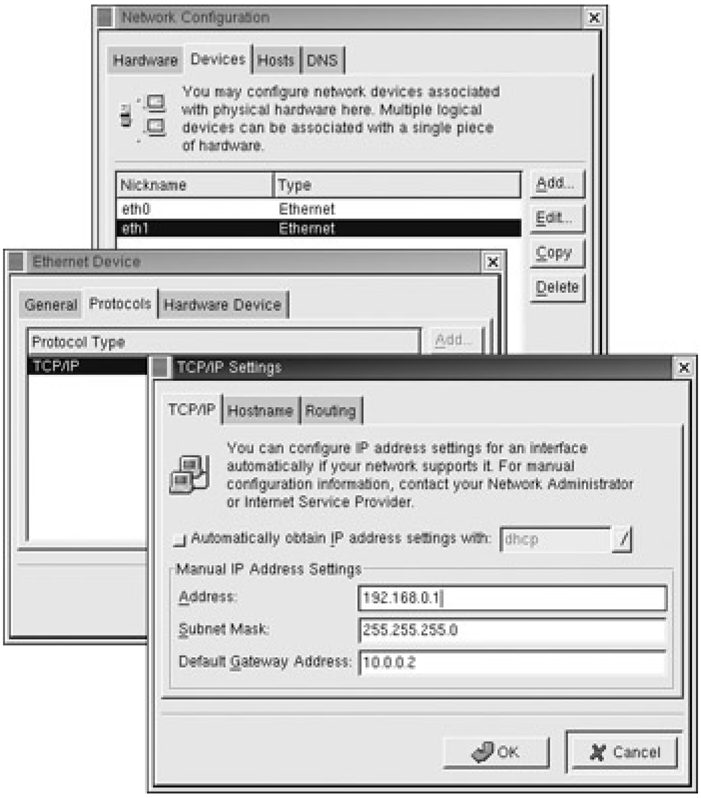
Hostname: turtle.mytrek.com

Aliases: turtle

Figure 5-3: Network Configuration Hosts panel

Note If you are having trouble connecting with an Ethernet device, make sure the Hosts panel lists your hostname and IP address, not just localhost. If your hostname is not there, add it.

The Devices panel lists configured network devices on your system (see Figure 5-4). Making entries here performs the same function as ifconfig. An entry shows the device name and its type. Use the Add, Edit, Copy, and Delete buttons to manage the device entries. When you add or edit a device, you open a tabbed panel for configuring it, enabling you to specify its IP address, host name, gateway, and the hardware device it uses. For example, when you installed Red Hat, any Ethernet network devices you had installed would be listed here. Editing the device opens a configuration window with three tabbed panels: General, Protocols, and Hardware Device. The Hardware panel selects a hardware device to use from a list of installed devices. In the General panel, you can set features such as activation at boot time or edit the nick name. The Protocols panel will list the protocols used on this device, usually TCP/IP. Editing the protocol will open a TCP/IP Settings window with tabbed panels for TCP/IP, Hostname, and Routing. Here you can enter the IP address assigned to the device, along with its netmask and network gateway (see Figure 5-4). In the Hostname panel, you can enter the devices hostname. Should you add a new network device, you will need to use the Device panel and its Protocol and TCP/IP settings windows to assign the device an IP address, hostname, netmask, and gateway, among other features.



You may configure network devices associated with physical hardware here. Multiple logical devices can be associated with a single piece of hardware.

Nickname	Type
eth0	Ethernet
eth1	Ethernet

Protocol Type
TCP/IP

You can configure IP address settings for an interface automatically if your network supports it. For manual configuration information, contact your Network Administrator or Internet Service Provider.

□ Automatically obtain IP address settings with: dhcp

Manual IP Address Settings

Address: 192.168.0.1

Subnet Mask: 255.255.255.0

Default Gateway Address: 10.0.0.2

Figure 5-4: Network Configuration Devices

When you finish and are ready to save your configuration, click the Apply button to have your changes take effect. If you want to abandon the changes you made, you can close without saving. You can run Network Configuration at any time to make changes in your network configuration.

You can also use Network Configuration to configure a PPP device for a modem. When you click Add and select modem as the interface, a Modem Dialup Configuration window opens with several panels including Provider, Options, and Protocol. Select the Provider panel to display entries for your ISP's dial-up phone number as well as your login name and password. On the Options panel, you can set PPP options (see Chapter 36). In the Protocol's TCP/IP entry, you can elect to have your DNS information, such as your hostname and name servers, obtained automatically from the provider.

Network Configuration with Linuxconf and Webmin

To configure a LAN connection in Linuxconf, start netconf from a terminal window. This displays a window with buttons for various network configuration options. Click the Basic Host Information button to display the host configuration window. In the first Adapter panel, you can enter the IP address, network device (usually eth0), and the kernel module to use (the drivers for your Ethernet card) along with other data like the hostname and netmask. Then, in the Network Configurator window, click the Name Server Specification button to display the Resolver Configuration window, where you can enter the IP addresses for the domain name servers on your network (see Figure 5-3). Click the Routers and Gateway button to enter the IP address for the gateway computer.

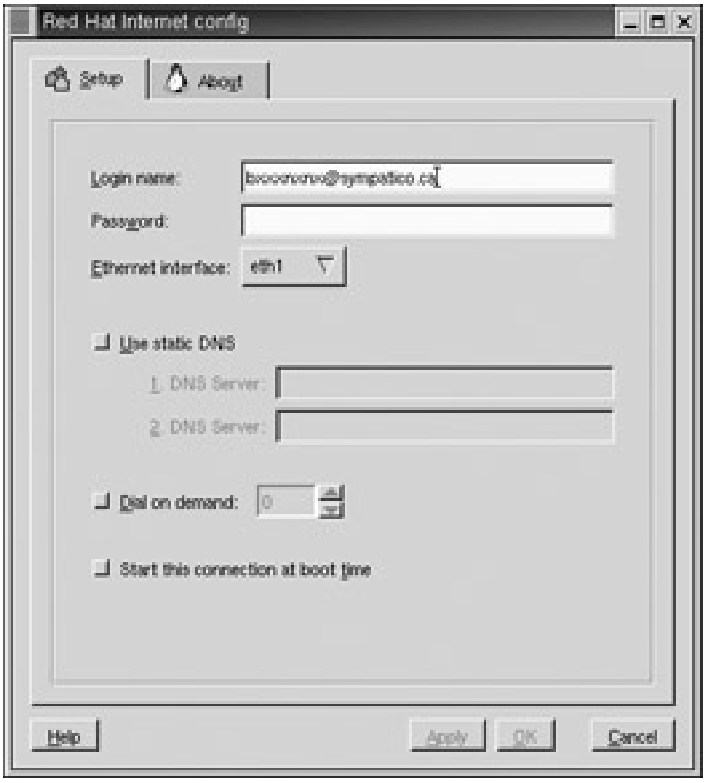
To use netconf to configure PPP connections, click the PPP/SLIP/PLIP button on the Network Configurator window. A window then opens that asks you to choose the type of interface you want. Select PPP. Then a small window opens displaying a ppp0 entry. Double-click it to display the PPP interface window with panels for setting your modem connections, the phone number to dial, and the Expect and Send entries for your login name and password. To activate a connection, click Connect. You can also use dial-up managers like Kppp and gnomeppp to set up and manage your PPP connections.

With Webmin, select the Network Configuration page on the Hardware page. From the Network Configuration page, select the Network Interfaces page to configure your Ethernet device, entering information like the IP address, the netmask, and the device name (see Figure 5-4). On the Routing and Gateways page, you enter the IP address of your network's gateway, and on the DNS page you enter your name server addresses. On the Host Address page, you enter the hostname and IP address for your system along with any others you want.

DSL and ISDN

To connect using DSL you use adsl-config, and for ISDN connections you use isdn-config. Both can be accessed through internet-config on the System menu. This will open a dialog box where you can select A-DSL/T-DSL, ISDN, or modem connections. The modem connection starts up Kppp, which is described in Chapter 36. You can also start them independently by entering their command in a terminal window.

adsl-config will display a dialog labeled Red Hat Internet Configuration. There will be entries for entering your login name, password, and the Ethernet interface your DSL modem is attached to (see Figure 5-5). You will also need to enter the IP addresses for the DNS servers provided by your ISP. You can also elect to have the connection automatically made up when your system starts up.



Login name: [illegible]@sympatico.ca

Password:

Ethernet interface: eth1

□ Use static DNS

1. DNS Server:

2. DNS Server:

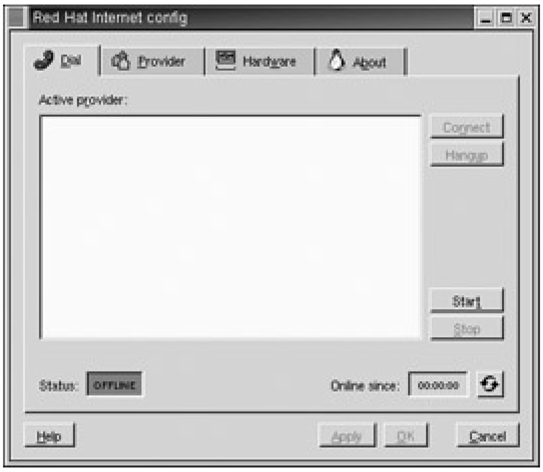
□ Dial on demand: 0

□ Start this connection at boot time

Figure 5-5: adsl-config, configuring DSL connections

adsl-config makes use of the pppoe utility to make your DSL connections. pppoe enables the use of dynamic IP addresses with an ISP over a DSL connection. Red Hat Linux uses the Roaring Penguin package of pppoe commands (rp-pppoe). As an alternative to adsl-config, you can use the **adsl-setup** command to configure your DSL connection. adsl has a command line interface and can be run at any shell prompt. As with adsl-config, you are prompted to enter your user name, password, Ethernet card, and domain name server addresses. You can also specify basic firewall security levels. You can then establish your DSL connection with the **adsl-start** command, and disconnect with the **adsl-stop** command.

isdn-config will display a dialog labeled Red Hat ISDN config and showing four panes labeled Dial, Provider, Hardware, and About (see Figure 5-6). On the Dial pane you can make a connection to a selected ISP. On the Provider pane you enter information about your ISP. You use the Hardware pane to configure your ISDN modem.



Active provider:

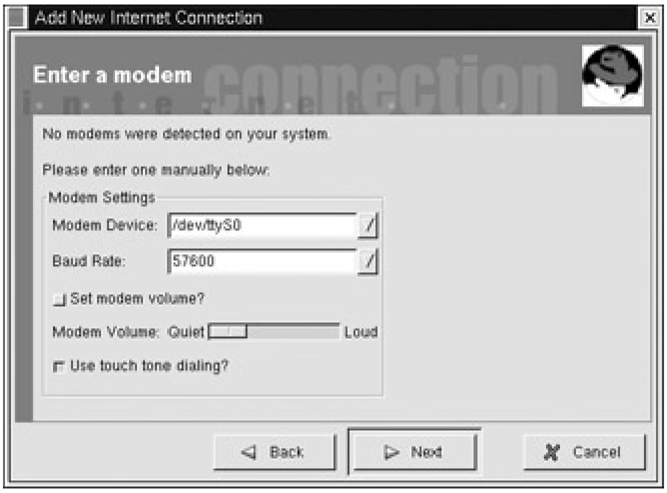
Status: OFFLINE

Online since: 00:00:00

Figure 5-6: isdn-config, configuring ISDN connections

The Red Hat PPP Dialer

If you have a dial-up connection to an Internet service provider (ISP), you need to configure a PPP interface. Almost all ISPs currently use PPP connections. You can easily set up a PPP connection using the Red Hat PPP Dialer (rp3). Select its entry in the Internet submenu on the Gnome desktop. If you do not have any Internet connections set up already, the Add New Internet Connection dialog box starts up (to add a new connection, you can click the New button on the Red Hat Dialup Configuration Tool window, opened from its entry on the Gnome Internet menu). If you have not yet configured your modem, the dialer automatically attempts to detect your modem and to provide information, such as the speed and serial device it uses. You see a screen displaying this information, along with the sound level, which you can adjust (see Figure 5-7).



Add New Internet Connection

Enter a modem

No modems were detected on your system.

Please enter one manually below:

Modem Settings

Modem Device: /dev/ttyS0

Baud Rate: 57600

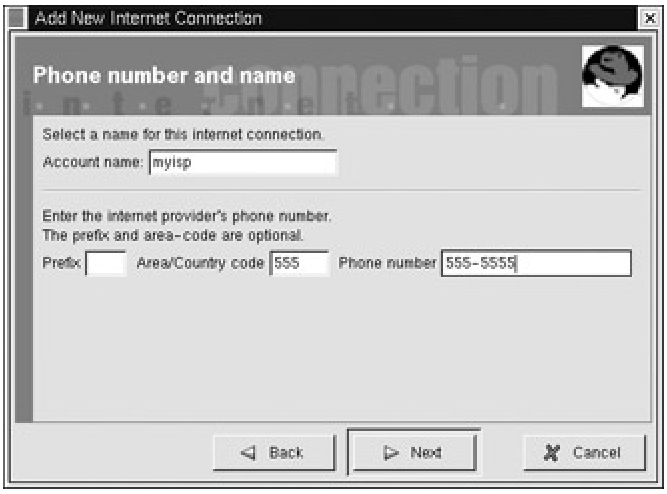
□ Set modem volume?

Modem Volume: Quiet ▭ Loud

☑ Use touch tone dialing?

Figure 5-7: Red Hat PPP Dialer modem configuration

Next, you are asked to enter the name you want to use to identify this connection on your system, the account name (see Figure 5-8). You also enter the phone number used to dial your ISP.



Add New Internet Connection

Phone number and name

Select a name for this internet connection.

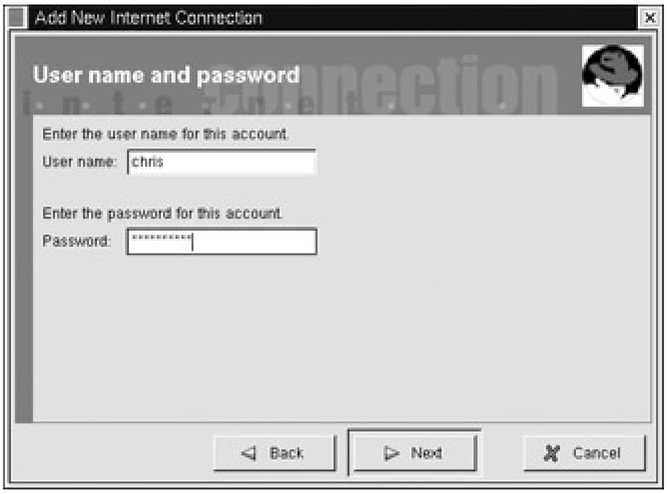
Account name: myisp

Enter the internet provider's phone number.
The prefix and area-code are optional.

Prefix: Area/Country code: 555 Phone number: 555-5555

Figure 5-8: Red Hat PPP Dialer name and phone number

You then enter the user name for your ISP account, along with its password (see Figure 5-9).



Enter the user name for this account.

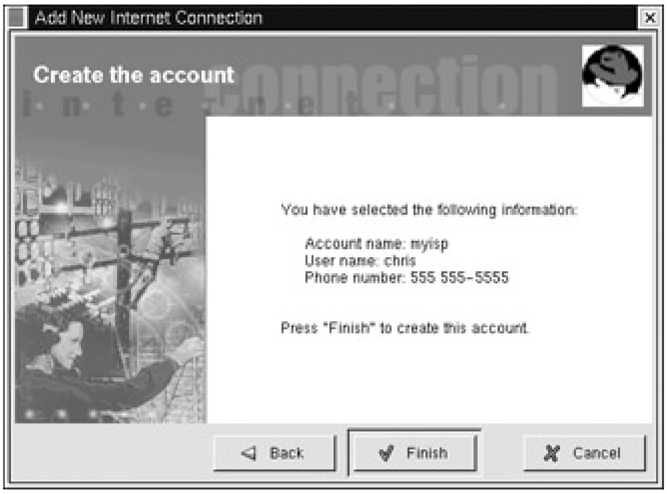
User name: chris

Enter the password for this account.

Password: **********

Figure 5-9: Red Hat PPP Dialer user name and password

When you finish, the final screen appears listing the account name, user name, and phone number, as shown in Figure 5-10.



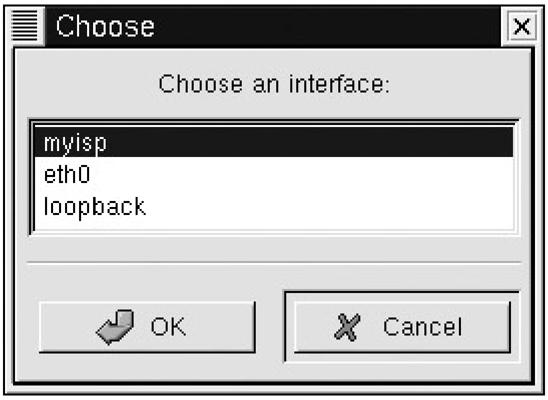
You have selected the following information:

Account name: myisp
User name: chris
Phone number: 555 555-5555

Press "Finish" to create this account.

Figure 5-10: Red Hat PPP Dialer final setup screen

Each time you want to connect, select the Red Hat PPP Dialer entry from the Gnome Internet menu. This displays a Choose window listing all your network connections, including any you created with the Red Hat PPP Dialer (see Figure 5-11). Double-click the account name you set up for your PPP connection. In Figure 5-11, the account name used for the PPP connection is myisp, which also shows an Ethernet connection (eth0) and the localhost connection.



Choose

Choose an interface:

Figure 5-11: Red Hat PPP Dialer

You are then prompted as to whether you want to start up the connection. Click the Yes button. A window appears that monitors the connection, showing a graph indicating the current activity on it (See Figure 5-12). You can minimize the monitor, docking it to the Gnome panel. Right-clicking the Monitor icon displays a menu with entries for starting, stopping, and configuring the connections, along with the connection properties. The Properties dialog box enables you to set such features as calculating the time and the cost of a connected session. The monitor is actually part of the Red Hat Network Monitor tool. You can select this tool independently on the Gnome Internet menu and display monitors for all your network connections.

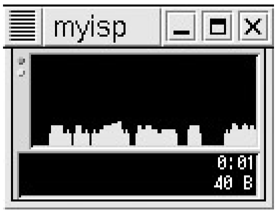


Figure 5-12: Red Hat Network Monitor tool

You can also place the network monitor for your PPP connection on the Gnome panel. Here it is displayed in a smaller size as a Gnome applet. To start your PPP connection, you only need to double-click the PPP network monitor. Right-clicking the monitor image displays a menu with options for starting and configuring your connection. To add a monitor to the Gnome panel, use the Panel menu on the Gnome Main menu. In the Panel menu, select Add Applet, and then the Network submenu, and, from there, select the RH PPP Dialer entry. This displays a list of all the network connections on your system. Select the one you want placed on the panel. The monitor bears the name you have to the PPP connection. In Figure 5-13, the PPP network monitor has the name myisp. You can place any of your network monitors on the Gnome panel. Figure 5-13 shows a monitor for both the Ethernet and PPP connections.

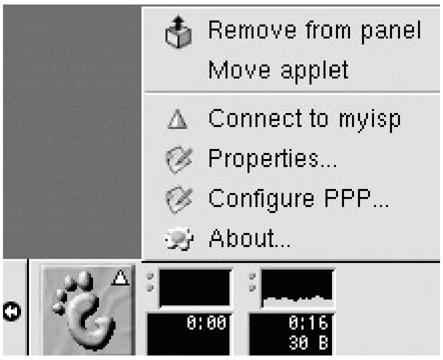
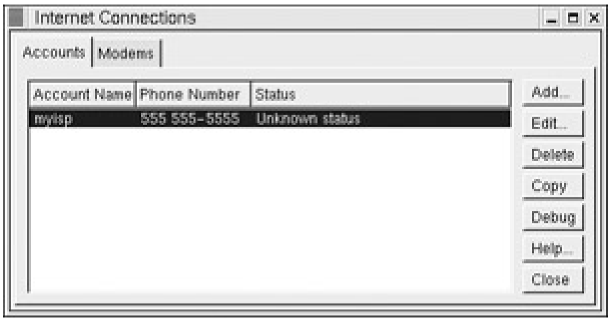


Figure 5-13: Red Hat network monitors on the panel

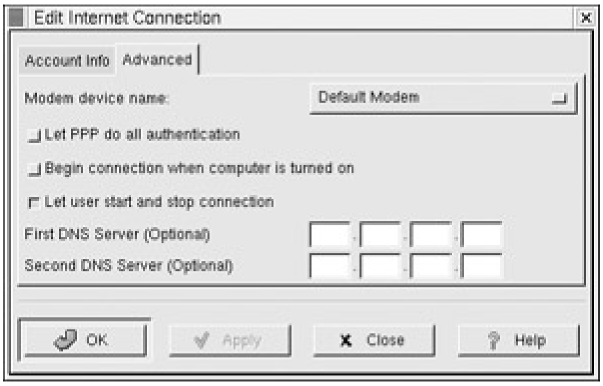
To change your setting or to add a new connection, use the Red Hat PPP Dialer Configuration tool. On the Gnome Internet menu, select the Dialup Configuration Tool entry to start up the Configuration tool. It then displays a window with two tabbed panels: one listing your accounts and the other for your modem configuration. Buttons on the right side of the Accounts panel enable you to add new connections or to edit current ones (see Figure 5-14).



Account Name	Phone Number	Status
myisp	555 555-5555	Unknown status

Figure 5-14: Red Hat PPP Dialer Configuration tool

The Edit button opens a window with panels for modifying connection and modem information (see Figure 5-15). On the Account Info panel, you can change your login name (user name), password, and phone number for your ISP. On the Advanced panel, you can enter information such as the IP address of your ISP's domain name servers.



Modem device name: Default Modem

□ Let PPP do all authentication

□ Begin connection when computer is turned on

☑ Let user start and stop connection

First DNS Server (Optional)

Second DNS Server (Optional)

Figure 5-15: Editing a PPP connection

The Red Hat PPP Dialer uses the wvdial utility to perform the connection operations. You can find the configuration information for your connections in the **/etc/wvdial.conf** file. You can, if you choose, edit this file directly to configure your PPP connections.

Command Line PPP Access: wvdial

If, for some reason, you have been unable to set up your X Window System, you may have to set up such a network connection from the command line interface instead of a desktop. The following discussion shows how to make such a connection using wvdial. wvdial is the standard dialer used on Red Hat systems.

For a dial-up PPP connection you can use the wvdial dialer, the same dialer used for the Red Hat PPP Dialer. wvdial is an intelligent dialer, which not only dials up an ISP service but also performs login operations, supplying your user name and password. wvdial first loads its configuration from the **/etc/wvdial.conf** file. In here, you can place modem and account information, including modem speed and serial device, as well as ISP phone number, user name, and password. The **wvdial.conf** file is organized into sections, beginning with a section label enclosed in brackets. A section holds variables for different parameters that are assigned values, such as username = chris. The default section holds default values inherited by other sections, so you needn't repeat them. Table 5-2 lists the wvdial variables.

Table 5-2: wvdial Variables

Variable	**Description**
Inherits	Explicitly inherit from the specified section. By default, sections inherit from the [Dialer Defaults] section.
Modem	The device wvdial should use as your modem. The default is **/dev/modem**.
Baud	The speed at which wvdial communicates with your modem. The default is 57,600 baud.
Init1 ... Init9	Specifies the initialization strings to be used by your modem. wvdial can use up to 9. The default is "ATZ" for Init1.
Phone	The phone number you want wvdial to dial.
Area Code	Specifies the area code, if any.
Dial Prefix	Specifies any needed dialing prefix-for example, 70 to disable call waiting or 9 for an outside line.
Dial Command	Specifies the dial operation. The default is "ATDT".
Login	Specifies the user name you use at your ISP.
Login Prompt	If your ISP has an unusual login prompt, you can specify it here.
Password	Specifies the password you use at your ISP.
Password Prompt	If your ISP has an unusual password prompt, you can specify it here.
PPPD PATH	If pppd is installed on your Linux system somewhere other than **/usr/sbin/pppd**, you need to specify its location with this option.
Force Address	Specifies a static IP address to use (for ISPs that provide static IP addresses to users).
Remote Name	For PAP or CHAP authentication, you may have to change this to your ISP's authentication name. The default value is *.
Carrier Check	Setting this option to No disables the carrier check by your

Table 5-2: wvdial Variables

Variable	Description
	modem. Used for a modem that reports its carrier line is always down.
Stupid Mode	In Stupid Mode, wvduak does not attempt to interpret any prompts from the terminal server and starts pppd after the modem connects.
New PPPD	Enable this option for use with pppd version 2.3.0 or newer. Instructs pppd to look for a required file **/etc/ppp/peers/wvdial**.
Default Reply	Specifies the default response for prompts that wvdial does not recognize. The default is ppp.
Auto Reconnect	If enabled, wvdial attempts to reestablish a connection automatically if you are randomly disconnected by the other side. This option is on by default.

You can use the wvdialconf utility to create a default **wvdial.conf** file for you automatically. wvdialconf will detect your modem and set default values for basic features. You can then edit the **wvdial.conf** file and modify the Phone, Username, and Password entries with your ISP dial-up information. Remove the preceding **;** to unquote the entry. Any line beginning with a **;** is ignored as a comment.

```
$ wvdialconf
```

You can also create a named dialer, such as *myisp* in the following example. This is helpful if you have different ISPs you log into. The following example shows the **/etc/wvdial.conf** file:

/etc/wvdial.conf

```
[Modem0]
Modem = /dev/ttyS0
Baud = 57600
Init1 = ATZ
SetVolume = 0
Dial Command = ATDT

[Dialer Defaults]
Modem = /dev/ttyS0
Baud = 57600
Init1 = ATZ
SetVolume = 0
Dial Command = ATDT

[Dialer myisp]
Username = chris
Password = mypassword
Modem = /dev/ttyS0
Phone = 555-5555
Area Code = 555
Baud = 57600
Stupid mode = 0
```

To start wvdial, enter the command **wvdial**, which then reads the connection configuration information from the **/etc/wvdial.conf** file. wvdial then dials the ISP and initiates the PPP connection, providing your user name and password when requested.

```
$ wvdial
```

You can set up connection configurations for any number of connections in the **/etc/wvdial.conf** file. To select one, enter its label as an argument to the **wvdial** command, as shown here:

```
$ wvdial myisp
```

Modem Setup

If you have a modem connected to your PC, it is connected to one of four communications ports. The PC names for these ports are COM1, COM2, COM3, and COM4. These ports can also be used for other serial devices, such as a serial mouse (though not for PS/2 mice). Usually, a serial mouse is connected to COM1 and a modem is connected to COM2, though in many cases your modem may be connected to COM4. Find out which ports your modem and mouse are connected to, because you must know this to access your modem. On the PC, COM1 and COM3 share the same access point to your computer; the same is true of COM2 and COM4. For this reason, if you have a serial mouse connected to COM1, you should not have your modem on COM3. You could find your mouse cutting out whenever you use your modem. If your mouse is on COM1, your modem should either be on COM2 or COM4.

In Linux, you use the serial communication ports for your modem. Serial ports begin with the name **/dev/ttyS,** with an attached number from 0 to 3. (Notice the numbering begins from 0, not 1.) The first port, COM1, is **/dev/ttyS0**, and **/dev/ttyS1** is the second port. The third and fourth ports are **/dev/ttyS2** and **/dev/ttyS3**. In many Linux communication programs, you need to know the port for your modem, which is either **/dev/ttyS1** for COM2 or **/dev/ttyS3** for COM4.

Some communication programs try to access the modem port using only the name **/dev/modem**. This is meant to be an alias, another name, for whatever your modem port actually is. If your system has not already set up this alias, you can easily create this alias using the **ln -s** command or the modemtool utility on Red Hat. modemtool has a GUI interface and is run on an X Window System desktop, such as Gnome or KDE. It displays four entries, one for each serial port. Click the one that applies to your system.

You can also create an alias on the command line using the **ln** command. The following example creates an alias called **modem** for the COM2 port, **/dev/ttyS1**. If your modem port is **/dev/ttyS3,** use that instead. (You must be logged in as a root user to execute this command.) The following example sets up the **/dev/modem** alias for the second serial port, **/dev/ ttyS1**:

```
# ln -s /dev/Stty1 /dev/modem
```

Your **/dev/mouse** alias should already be set up for the port it uses. For a serial mouse, this is usually the COM1 port, **/dev/ ttyS0**. If the alias is not set up or if you need to change it, you can use the **ln -s** command. The following example sets up the **/dev/mouse** alias for the first serial port, **/dev/ ttyS0**.

```
# ln -s /dev/ttyS0 /dev/mouse
```

Chapter 6: Security Configuration

Overview

Once you have installed your Linux system, you should carry out some basic security measures to protect your system from outside attacks. Systems connected to the Internet are open to attempts by outside users to gain unauthorized access. This usually takes the following forms:

- Trying to break into the system
- Having broken in, changing or replacing system files with hacked or corrupt versions
- Attempting to intercept communications from remote users
- Changing or replacing messages sent to or from users
- Pretending to be a valid user

Firewalls, intrusion protection, encryption, data integrity, and authentication are ways of protecting against such attacks (see Chapter 40 also).

- A firewall prevents any direct unauthorized attempts at access.
- Intrusion protection checks the state of your system files to see if they have been tampered with by someone who has broken in.
- Encryption protects transmissions by authorized remote users, providing privacy.
- Integrity checks like modification digests guarantee that messages and data have not been intercepted and changed or substituted en route.
- Authentication methods such as digital signatures can verify that the user claiming to send a message or access your system is actually that person.

You can use encryption, integrity checks, and authentication to protect both messages you send as e-mail or files you attach. The GNU Privacy Guard encryption package lets you encrypt your e-mail messages or files you want to send, as well as letting you sign them with an encrypted digital signature authenticating that the message was sent by you. The digital signature also includes encrypted modification digest information that provides an integrity check, allowing the recipient to verify that the message received is the original and not one that has been changed or substituted.

You will also need to check the integrity of your system to make sure that it has not already been broken into. With the Tripwire intrusion detection software, you can take a snapshot of your system, taking note of different features for critical files like size and permissions of configuration files. Later, you can check the current state of those critical files with your previous snapshot version to see if they have changed in any way. If they have, it may be evidence that an intruder has entered your system and is changing files.

A good foundation for your network security is to set up a Linux system to operate as a firewall for your network, protecting it from unauthorized access. You can use a firewall to implement either packet filtering or proxies. *Packet filtering* is simply the process of deciding whether a packet received by the firewall host should be passed on into the local network. It checks the address of the packet and sends the packet on, if it's allowed. The firewall package currently in use for Red Hat is Netfilter (iptables). Older releases of Red Hat (kernel 2.2 and below) use an earlier version called ipchains. To implement a firewall, you simply provide a series of rules to govern what kind of access you want to allow on your system. If that system is also a gateway for a private network, the system's firewall capability can effectively protect the network from outside attacks.

Another way to protect access to your system is to provide secure user authentication with encrypted passwords, a Lightweight Directory Access Protocol (LDAP) service, and Pluggable Authentication Modules (PAM). These are discussed in detail in Chapter 30. User authentication can further be controlled for certain services by Kerberos servers, discussed in Chapter 40.

To protect remote connections from hosts outside your network, transmissions can be encrypted. For Linux systems, you can use the Secure Shell (SSH) suite of programs to encrypt any transmissions, preventing them from being read by anyone else. If you don't use SSH, it is best to avoid the standard remote communications tools such as telnet and rcp (see Chapter 21) for remote access over an unprotected networks like the Internet. Outside users may also try to gain unauthorized access through any Internet services you may be hosting, such as a Web site. In such a case, you can set up a proxy to protect your site from attack. For Linux systems, use Squid proxy software to set up a proxy to protect your Web server (see Chapter 27).

This chapter will show you some simple steps you can take to provide a basic level of security. The GNU Privacy Guard encryption and Tripwire intrusion detection software are covered in detail. Netfilter, Squid, and SSH are covered briefly (see Chapters 27 and 40 for a more detailed analysis of these applications). Table 6-1 lists security applications used on Red Hat systems.

Note Numerous older security applications are also available for Linux such as COPS (Computer Oracle and Password System) to check password security; Tiger, which scans your system for unusual or unprotected files; and SATAN (Security Administration Tool for Analyzing Networks), which checks your system for security holes. Crack is a newer password auditing tool that you can use to check how well your password security performs under dictionary attacks.

GNU Privacy Guard: Encryption and Authentication

To protect messages that you send by e-mail, Red Hat provides GNU Privacy Guard (GnuPG) encryption and authentication. GnuPG is GNU open source software that works much like Pretty Good Privacy (pgp) encryption. With GnuPG, you can both encrypt your messages and digitally sign them-protecting the message and authenticating that it is from you. Currently, KMail and exmh both support GnuPG encryption and authentication. On KMail, you can select the encryption to use on the Security panel in the Options window.

Table 6-1: Security Applications

Applications	Description
GNU Privacy Guard (GPG)	Encryption and digital signatures (Chapter 6)
Tripwire	Intrusion detection (Chapter 6)

Table 6-1: Security Applications

Applications	Description
Netfilter (iptables and lokkit)	Firewall packet filtering (Chapter 40)
Squid	Web proxy server (Chapter 27)
SSH	Secure Shell encryption and authentication for remote access (Chapter 40)
Kerberos	User authentication for access to services (Chapter 40)
Pluggable Authorization Modules (PAM)	Authentication management and configuration (Chapter 30)
Shadow passwords	Password encryption (Chapter 30)
Lightweight Directory Access Protocol (LDAP)	User management and authorization (Chapter 30)

Public-Key Encryption and Digital Signatures

GnuPG makes use of public-key cryptography to encrypt data. Public-key encryption uses two keys to encrypt and decrypt a message, a private key and a public key. The private key you always keep and use to decrypt messages you have received. The public key you make available to those you send messages to. They then use your public key to encrypt any message they want to send to you. The private key decrypts messages, and the public key encrypts them. Each user has private and public keys. Reciprocally, if you want to send messages to another user, you would first obtain the user's public key and use it to encrypt the message you want to send to the user. The user then decrypts the messages with his or her own private key. In other words, your public key is used by others to encrypt the messages you receive, and you use other user's public keys to send messages to them. Each user on your Red Hat system can have their own public and private keys. They will use the gpg program to generate them and keep their private key in their own directory.

A digital signature is used to both authenticate a message and provide a integrity check. Authentication guarantees that the message has not been modified-that it is the original message sent by you-and the integrity check verifies that it has not been changed. Though usually combined with encrypted messages to provide a greater level of security, digital signatures can also be used for messages that can be sent in the clear. For example, you would want to know if a public notice of upgrades of a Red Hat release was actually sent by Red Hat, and not by someone trying to spread confusion. Such a message still needs to be authenticated, checked to see if it was actually sent by the sender or, if sent by the original sender, was not somehow changed en route. Verification like this protects against modification or substitution of the message by someone pretending to be the sender.

Digitally signing a message involves generating a checksum value from the contents of the message using an encryption algorithm such as the MD5 modification digest algorithm. This is a unique value that accurately represents the size and contents of your message. Any changes to the message of any kind would generate a different value. Such a value provides a way to check the integrity of the data. The MD5 value is then itself encrypted with your private key. When the user receives your message, they decrypt your digital signature with your public key. The user then generates an MD5 value of the message received and compares it with the MD5 value you sent. If they are the same, the message is authenticated-it is the original message sent by you, not a false one sent by a user pretending to be you. The user can use GnuPG to decrypt and check digital signatures.

Normally, digital signatures are combined with encryption to provide a more secure level of transmission. The message would be encrypted with the recipient's public key, and the digital signature encrypted with your private key. The user would decrypt both the message (with their own private key) and then the signature (with your public key). They would then compare the signature with one the user generates from the message to authenticate it. When GnuPG decodes a message, it will also decode and check a digital signature automatically. Figure 6-1 shows the process for encrypting and digitally signing a message.

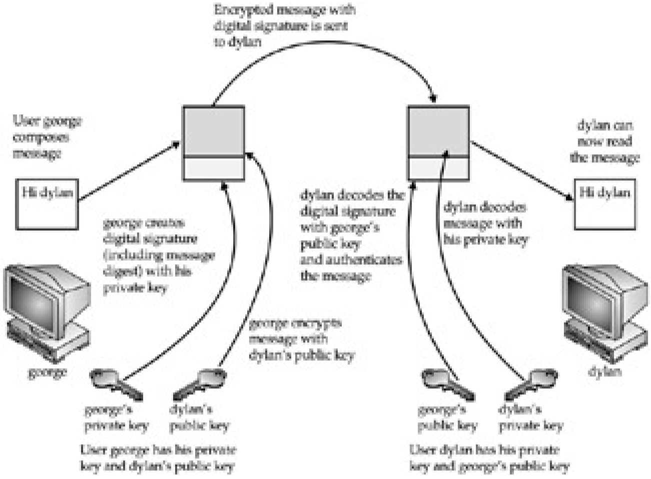

Encrypted message with digital signature is sent to dylan
User george composes message
Hi dylan
george creates digital signature (including message digest) with his private key
george
george encrypts message with dylan's public key
george's private key
dylan's public key
User george has his private key and dylan's public key
dylan decodes the digital signature with george's public key and authenticates the message
dylan decodes message with his private key
dylan can now read the message
Hi dylan
dylan
george's public key
dylan's private key
User dylan has his private key and george's public key

Figure 6-1: Public-key encryption and digital signatures

GPG operations are carried out with the gpg command, which uses both commands and option to perform tasks. Commonly used commands and options are listed in Table 6-2. Some commands and options have a short form that use only one dash. Normally two are used.

GnuPG Setup:gpg

Before you can use GnuPG on Red Hat, you will have generate your private and public keys (see the Red Hat Customization Guide for details). On the command line (terminal window) enter the **gpg** command with the **--gen-key** command. The gpg program will then prompt with different options for creating your private and public keys. You can check the gpg Man page for information on using the gpg program.

```
gpg --gen-key
```

You are first asked to select the kind of key you want. Normally, you would just select the default entry, which you can do by just pressing the ENTER key. Then you choose the key size, usually the default 1024. You then specify how long the key is to be valid-usually there is no expiration. You will then be asked to enter a user ID, comment, and e-mail address. Press ENTER to then be prompted for each in turn. These elements identify the key, any of which can be used as the key's name. You use the key name when performing certain GPG tasks like signing a key or creating a revocation certificate. For example, the following elements create a key for the user richlp with the comment "author" and the e-mail address **richlp@turtle.mytrek.com**.

Table 6-2: GPG Commands and Options

GPG Commands	**Description**
-s, **--sign**	Signs a document, creating a signature. May be combined with **--encrypt**.
--clearsign	Creates a clear text signature.
-b, --detach-sign	Creates a detached signature.
-e, --encrypt	Encrypts data. May be combined with **--sign**.
--decrypt [*file***]**	Decrypts file (or stdin if no file is specified) and writes it to stdout (or the file specified with **--output**). If the decrypted file is signed, the signature is verified.
--verify [[*sigfile***] [***signed-files***]]**	Verifies a signed file. The signature can either be contained with the file or be a separate detached signature file.
--list-keys [*names***]**	Lists all keys from the keyrings or those specified.
--list-public-keys [*names***]**	Lists all keys from the public keyrings or those specified.
--list-secret-keys [*names***]**	Lists your private (secret) keys.
--list-sigs [*names***]**	Lists your keys along with any signatures they have.
--check-sigs [*names***]**	Lists keys and their signatures and verifies the signatures.
--fingerprint [*names***]**	Lists fingerprints for specified keys.
--gen-key	Generates a new set of private and public keys.
--edit-key *name*	Edits your keys. Use commands to perform most key operations such as sign to sign a key or password to change your passphrase.
--sign-key *name*	Signs a public key with your private key. Same as sign in **--edit-key**.
--delete-key *name*	Removes a public key from the public keyring.
--delete-secret-key *name*	Removes private and public key from both the secret and public keyrings.
--gen-revoke	Generates a revocation certificate for your own key.
--export [*names***]**	Exports a specified key from your keyring. With no arguments, exports all keys.
--send-keys [*names***]**	Exports and sends specified keys to a keyserver. The option **--keyserver** must be used to give the name of this keyserver.
--import [*files***]**	Imports keys contained in files into your public keyring.
-a, --armor	Creates ASCII armored output, ASCII version of encrypted data.
-o, --output *file*	Writes output to a specified file.
--default-key *name*	Specifies the default private key to use for signatures.

Table 6-2: GPG Commands and Options

GPG Commands	Description
--keyserver *site*	The keyserver to look up public keys not on your keyring. Can also specify the site to send your public key to. **host -l pgp.net | grep www.keys** will list the keyservers.
-r, --recipient *names*	Encrypts data for the specified user, using that user's public key.
--default-recipient *names*	Specifies the default recipient to use for encrypting data.

```
Richard Petersen (author) <richlp@turtle.mytrek.com>
```

You can use any unique part of a key's identity to reference that key. For example, the string "Richard" would reference the above key, provided there are no other keys that have the string "Richard" in them. "richlp" would also reference the key, as would "author". Where a string matches more than one key, all the matched ones would be referenced.

gpg will then ask you to enter a passphrase, used to protect your private key. Be sure to use a real phrase, including spaces, not just a password. gpg then generates your public and private keys and places them in the **.gnupg** directory. The private key is kept in a file called the **secring.gpg** in your **.gnupg** directory. The public key is placed in the **pubring.gpg** file, to which you can add the public keys of other users. You can list these keys with the **--list-keys** command.

In case you later need to change your keys, you can create a revocation certificate to notify others that the public key is no longer valid. For example, if you forget your password or someone else discovers it, you can use the revocation certificate to tell others that your public key should no longer be used. In the next example, the user creates a revocation certificate for the key richlp and places it in the file **myrevoke.asc**:

```
gpg --output myrevoke.asc --gen-revoke richlp
```

For other users to decrypt your messages, you have to make your public key available to them. They, in turn, have to send you their public keys so that you can decrypt any messages you receive from them. In effect, enabling encrypted communications between users involves all of them exchanging their public keys. The public keys then have to be verified and signed by each user that receives them. The public keys can then be trusted to safely decrypt messages.

If you are sending messages to just a few users, you can manually e-mail them your public key. For general public use, you can post your public key on a key server, which anyone can then download and use to decrypt any message they receive from you. The OpenPGP Public Keyserver is located at www.keyserver.net. You can send directly to the key server with the **-keyserver** option and **--send-key** command. The **send-key** command takes as its argument your e-mail address. You only need to send to one keyserver, as it will share your key with other key servers automatically.

```
gpg --keyserver search.keyserver.net --send-key chris@turtle.mytrek.com
```

If you want to send your key directly to another user, you will should generate an armored text version of the key that you can then e-mail. You do this with the **--armor** and **--export** options, using the **--output** option to specify a file to place the key in. The **--armor** option will generate an ASCII text version of the encrypted file so that it can be e-mailed directly, instead of as an attached binary. Files that hold an ASCII encoded version of the encryption normally have the extension **.asc**, by convention. Binary encrypted files normally use the extension **.gpg**. You can then e-mail the file to users to whom you want to send encrypted messages.

```
# gpg --armor --export richlp@turtle.mytrek.com --output richlp.asc
# mail -s 'mypubkey' george@rabbit.mytrek.com < richlp.asc
```

Many companies and institutions post their public key files on their Web sites where they can be downloaded and used to verify encrypted software downloads or official announcements.

Note Some commands and options for GPG have both a long and short form. For example the **--armour** command can be written as **-a**, **--output** as **-o**, **--sign** as **-s**, and **--encrypt** as **-e**. Most others, like **--export**, have no short form.

To decode messages from other users, you will need to have their public keys. They can either send them to you or you can download them from a key server. Save the message or Web page containing the public key to a file. You will then need to import, verify, and sign the key. Use the file you received to import the public key to your **pubring** file. In the following example, the user imports george's public key, which he has received as the file **georgekey.asc**.

```
gpg --import georgekey.asc
```

You should, for example, download the Red Hat public key, currently located at **http://www.redhat.com/about/contact.html**. Click on the Public Encryption Key link. From there you can access a page that displays just the public key. You can then save this page as a file and use that file to import the Red Hat public key to your keyring. In the following example, the user saved the page showing just the Red Hat public key as **myredhat.asc**, and then imported that file:

```
gpg --import myredhat.asc
```

Note You can remove any key, including your own private key, with the **--delete-key** and **--delete-secret-key** commands.

To manually check that a public key file was not modified in transit, you can check its fingerprint. This is a hash value generated from the contents of the key, much like a modification digest. Using the **--fingerprint** option you can generate a hash value from the key you installed, then contact the sender and ask them what the hash value should really be. If they are not the same, you know the key was tampered with in transit.

```
gpg --fingerprint george@rabbit
```

You do not have to check the fingerprint to have GPG operate. This is just an advisable precaution you can perform on your own. The point is that you need to be confident that the key you received is valid. Normally you can accept most keys from public servers or known sites like Red Hat as valid, although it is easy to check their posted fingerprints. Once assured of the key's validity, you can then sign it with your private key. Signing a key notifies GPG that you officially accept the key.

To sign a key you use the **gpg** command with the **--sign-key** command and the key's name.

```
gpg --sign-key george@rabbit
```

Alternatively, you can edit the key with the **--edit-key** command to start an interactive session in which you can enter the command **sign** to sign the key and **save** to save the change. Signing a key involves accessing your private key, so you will be prompted for your passphrase. Once finished, leave the interactive session with the **quit** command.

Normally, you would want to post a version of your public key that has been signed by one or more users. You can do the same for other users. Signing a public key provides a way to vouch for the validity of a key. It indicates that someone has already checked it out. Many different users could sign the same public key. For a key that you have received from another user, and that you have verified, you can sign and then return the signed version to that user. Once you have signed the key, you can generate a file containing the signed public version. You can then send this file to the user.

```
gpg -a --export george@rabbit --output  georgesig.asc
```

The user would then import the signed key and then export it to a key server.

Tip Should you want to start over from scratch, you can just erase your **.gnupg** directory, although this is a drastic measure-you lose any keys you have collected.

Using GnuPG

GnuPG encryption is currently supported on Red Hat by KMail and exmh mail clients. You can also use the GNU Privacy Assistant (GPA), a GUI front end, to manage gpg tasks. You can also use the **gpg** command to manually encode and decode messages, including digital signatures if you wish. As you perform GPG tasks you will need to reference the keys you have using their key names. Bear in mind that you only need a unique identifying substring to select the key you want. GPG performs a pattern search on the string you specify as the key name in any given command. If the string matches more than one key, all those matching will be selected. In the following example, the Sendmail string selects matches on the identities of two keys.

```
# gpg --list-keys "Sendmail"
pub  1024R/CC374F2D 2000-12-14
            Sendmail Signing Key/2001 <sendmail@Sendmail.ORG>
pub  1024R/E35C5635 1999-12-13
            Sendmail Signing Key/2000 <sendmail@Sendmail.ORG>
```

gpg provides several options for managing secure messages. The **e** option encrypts messages, the **a** option generates an armored text version, and the **s** option adds a digital signature. You will need to specify the recipient's public key, which you should already have imported into your **pubring** file. It is this key that is used to encrypt the message. The recipient will then be able to decode the message with their private key. Use the **--recipient** or **-r** options to specify the name of the recipient key. You can use any unique substring in the user's public key name.

The e-mail address usually suffices. You use the **d** option to decode received messages. In the following example, the user encrypts (**e**) and signs (**s**) a file generated in armored text format (**a**). The **-r** option indicates the recipient for the message (whose public key is used to enrypt the message).

```
gpg e -s -a -o myfile.asc -r george@rabbit.mytrek.com myfile
# mail george@rabbit.mytrek.com < myrile.asc
```

You can leave out the ASCII armour option if you want to send or transfer the file as a binary attachment. Without **--armour** or **-a** options, gpg generates an encoded binary file, not an encoded text file. A binary file can only be transmitted through e-mail as an attachment. As noted previously, ASCII armour versions usually have an extension of **.asc**, whereas binary version use **.gpg**.

When the other user receives the file, they can save it to a file named something like **myfile.asc**, and then decode the file with the **-d** option. The **-o** option will specify a file to save the decoded version in. GPG will automatically determine if it is a binary file or an ASCII armour version.

```
gpg -d -o myfile.txt myfile.asc
```

To check the digital signature of the file, you use the **gpg** command with the **--verify** option. This assumes that the sender has signed the file:

```
gpg --verify myfile.asc
```

However, you will need to have the signer's public key to decode and check the digital signature. If you do not, you will receive a message saying that the public key was not found. In this case, you will first have to obtain the signer's public key. You could access a key server that you think may have the public key, or request the public key directly from a Web site or from the signer. Then import the key as described previously.

You do not have to encrypt a file to sign it. A digital signature is a separate component. You can either combine the signature with a given file, or generate one separately. To combine a signature with a file you generate a new version that incorporates both. Use the **--sign** or **-s** commands to generate a version of the document that includes the digital signature. In the following example, the **mydoc** file is digitally signed with **mydoc.gpg** file containing both the original file and the signature.

```
gpg  -o mydoc.gpg  --sign mydoc
```

If, instead, you want to just generate a separate signature file, you use the **--detach-sig** command. The signature file usually has an extension like .sig. This has the advantage of not having to generate a complete copy of the original file. That file remains untouched. In the following example, the user creates a signature file called **mydoc2.sig** for the **mydoc2** file.

```
gpg -o mydoc2.sig --detach-sig mydoc2
```

To verify the file using a detached signature, the recipient user specifies both the signature file and the original file.

```
gpg --verify mydoc2.sig  mydoc2
```

You could also generate a clear sign signature to be used in text files. A clear sign signature is a text version of the signature that can be attached to a text file. The text file can be further edited by any text editor. Use the **--clearsign** option to create a clear signature. The following example creates a clear signed version of a text file called **mynotice.txt**.

```
gpg -o mysignotice.txt --clearsign mynotice.txt
```

Note Numerous GUI front ends and filters are available for GnuPG at www.gnupg.org. GPA (GNU Privacy Assistant) provides a Gnome-based front end to easily encrypt and decrypt files. You can select files to encode, choose the recipients (public keys to use), and add a digital signature if you wish. You can also use GPA to decode encoded files you receive. You can also manage your collection of public keys, the keys on your **keyring** file.

Tip Steganography is another form encryption that hides data in other kinds of objects such as images. You can use JPEG Hide and Seek software (JPHS) to encode and retrieve data in a JPEG image (jphide and jpseek). See **linux01.gwdg.de/~alatham/stego.html** for more details.

Checking Software Package Digital Signatures

One very effective use for digital signatures is to verify that a software package has not been tampered with. It is feasible that a software package could be intercepted in transmission and some of its system-level files changed or substituted. Software packages distributed by Red Hat, as well as those by reputable GNU and Linux projects, are digitally signed. The signature provides modification digest information with which to check the integrity of the package. The digital signature may be included with the package file or posted as a separate file. You use the **gpg** command with the **--verify** option to check the digital signature for a file.

First, however, you will need to make sure that you have the signer's public key. The digital signature was encrypted with the software distributor's private key. That distributor is the signer. Once you have that signer's public key, you can check any data you receive from them. In the case of a software distributor, once you have their public key, you can check any software they distribute. To obtain the public key, you can check a key server or, more likely, check their Web site. You can download the Red Hat public key from the Red Hat Web site at **http://www.redhat.com/about/contact.html**, as noted previously. Once you have downloaded the public key, you can add to your **keyring** with the **-import** option, specifying the name you gave to the downloaded key file (in this case, **myredhat.asc**):

```
# gpg --import redhat.asc
gpg: key CBA29BF9: public key imported
gpg: Total number processed: 1
gpg: imported: 1 (RSA: 1)
```

To download from a key server instead, you use the **--keyserver** option and the keyserver name.

You can use the **--fingerprint** option to check a key's validity if you wish. If you are confident that the key is valid, you can then sign it with the **-sign-key** command. In the following example, the user signs the Red Hat key, using the string "Red Hat" in the key's name to reference it. The user is also asked to enter his or her passphrase to allow use of his or her private key to sign the Red Hat public key.

```
# gpg --sign-key "Red Hat"
pub  1024R/CBA29BF9  created: 1996-02-20 expires: never  trust: -/q
(1). Red Hat Software, Inc. <redhat@redhat.com>
pub  1024R/CBA29BF9  created: 1996-02-20 expires: never  trust: -/q
 Fingerprint: 6D 9C BA DF D9 60 52 06  23 46 75 4E 73 4C FB 50
 Red Hat Software, Inc. <redhat@redhat.com>

Are you really sure that you want to sign this key
with your key: "Richard Petersen (author) <richlp@turtle.mytrek.com>"
Really sign? yes
You need a passphrase to unlock the secret key for
user: "Richard Petersen (author) <richlp@turtle.mytrek.com>"
1024-bit DSA key, ID 73F0A73C, created 2001-09-26
Enter passphrase:
#
```

Once you have the public key, you can check any RPM software packages for Red Hat with the **rpm** command and **-K** option. The following example checks the validity of the xcdroast and balsa software packages:

```
# rpm -K xcdroast-0.98alpha9-1.i386.rpm
xcdroast-0.98alpha9-1.i386.rpm: md5 OK
# rpm -K balsa-1.1.7-1.i386.rpm
balsa-1.1.7-1.i386.rpm: md5 OK
```

Many software packages in the form of compressed archives, **.tar.gz** or **tar.bz2**, will provide signatures in separate files that end with either the .asc or .sig extension. To check these, you use the **gpg** command with the **--verify** option. For example, the most recent Sendmail package is distributed in the form of a compressed archive, **.tar.gz**. Its digital signature is provided in a separate .sig file. First, you would download and install the public key for sendmail software obtained from the Sendmail Web site.

```
# gpg --import sendmail.asc
```

You should then sign the Sendmail public key that you just imported. In this example, the e-mail address was used for the key name.

```
gpg --sign-key sendmail@Sendmail.ORG
```

You could also check the fingerprint of the key for added verification.

You would then download both the compressed archive and the digital signature files. Decompress the **.gz** file to the **.tar** file with **gunzip**. Then, with the **gpg** command and the **--verify** option, use the digital signature in the **.sig** file to check the authenticity and integrity of the software compressed archive.

```
# gpg --verify sendmail.8.12.0.tar.sig sendmail.8.12.0.tar
gpg: Signature made Fri 07 Sep 2001 07:21:30 PM PDT using RSA key ID
CC374F2D
gpg: Good signature from "Sendmail Signing Key/2001
<sendmail@Sendmail.ORG>"
```

You could also just specify the signature file and gpg will automatically search for and select a file of the same name, but without the **.sig** or **.asc** extension.

```
# gpg --verify sendmail.8.12.0.tar.sig
```

In the future, when you download any software from the Sendmail site that uses this key, you just have to perform the **--verify** operation. Bear in mind though that different software packages from the same site may use different keys. You would have to make sure that you have imported and signed the appropriate key for the software you are checking.

Intrusion Detection: Tripwire

When someone breaks into a system, they will usually try to gain control by making their own changes to system administration files such as password files. They could create their own user and password information, allowing them access at any time, or simply change the root user password. They could also replace entire programs, such as the login program, with their own version. One method of detecting such actions is to use an integrity checking tool like Tripwire to detect any changes to system administration files. An integrity checking tool works by first creating a database of unique identifiers for each file or program to be checked. These can include features such as permissions and file size, but also, more importantly, checksum numbers generated by encryption algorithms from the file's contents. For example, in Tripwire, the default identifiers are checksum numbers created by algorithms like the MD5 modification digest algorithm and Snefru (Xerox secure hash algorithm). An encrypted value that provides such a unique identification of a file is known as a signature. In effect, a signature provides an accurate snapshot of the contents of a file. Files and programs are then periodically checked by generating their identifiers again and matching them with those in the database. Tripwire will generate signatures of the current files and programs and match them against the values previously generated for its database. Any differences are noted as changes to the file, and Tripwire then notifies you of the changes.

Note AIDE (Advanced Intrusion Detection Environment) is an alternative to Tripwire. It provides easy configuration and detailed reporting.

The Linux version of Tripwire is freely available as an open source product distributed under the GPL license. Tripwire also provides commercial versions for other operating systems. You can find out more about Tripwire at www.tripwire.com, and download the most recent release from www.tripwire.org. Tripwire is included with Red Hat and is discussed in the Red Hat Reference Manual. Detailed documentation is provided in a series of Man pages. **tripwire** discusses the **tripwire** command and its options. **twpolicy** describes in detail how Tripwire rules and directives work in the **twpol.txt** file. **twconfig** covers the configuration variables set in **twcfg.txt**. **twfiles** lists the different directories that Tripwire uses such as the **/var/lib/tripwire/report** directory that holds Tripwire check results. **twadmin** describes the usage of the **twadmin** command to create and display the policy (**tw.pol**) and configuration (**tw.cfg**) files.

Note Tripwire is not included in the Publisher's Edition. You can download it from the Red Hat FTP site.

You should install Tripwire when your system is in a secure state-as in not connected to any network. On Red Hat, Tripwire is installed as part of the standard installation. However, you should remain disconnected from a network after the installation while you configure and initialize Tripwire. Using Tripwire is a continual process of checking the Tripwire database for changes, making any configuration or policy changes that may be needed, and reinitializing the Tripwire database to reflect valid changes. The commands and files used in the Red Hat installation of Tripwire are listed in Table 6-3.

Note You can also check your log files for any suspicious activity. See Chapter 28 for a discussion on system logs. **/var/log/messages** in particular is helpful in checking for critical events such as user logins, FTP connections, and superuser logins.

Tripwire Configuration

To first use Tripwire, you will have to generate a configuration file and a policy file. These files are generated by the **twinstall.sh** script. If you just want to use the standard configuration, you can generate the files immediately by running the **twinstall.sh** script.

If you want to customize your configuration and policy files, you will have to first modify their editable versions in the **/etc/tripwire** directory. There are two versions of these files. One is a **.txt** file that you can edit to customize your configuration, and the other will be generated by **twinstall.sh** script using the **.txt** file. The configuration file will specify the Tripwire application directories and files, such as the directory where the Tripwire database is placed and reports are stored. **twcfg.txt** is the editable version of the configuration file. This file will already include the standard administrative files. You can edit this file to add any files of your own. The policy file holds the files, programs, and directories that you want Tripwire to check. The **twpol.txt** file is its editable version. You can edit this file to add or change policies to fit your system's particular needs. Once you have made the changes you want to the **twcfg.txt** and **twpol.txt** files, you can then use the **twinstall.sh** script to generate **the tw.cfg** and **tw.pol** files. These are the actually configuration and policy files that Tripwire uses, and, for security reasons, should never be touched.

Table 6-3: Tripwire Commands and Files

Commands and Files	Description
tripwire	Initialize and perform integrity checking.
twadmin	Administer Tripwire configuration and policy files, as well as Tripwire encryption keys.
twprint	Print and display Tripwire database and reports.
siggen	Generate new passphrases.
twinstall.sh	Generate keys and encrypted configuration and policy files.
/etc/tripwire/tw.cfg	Encrypted Tripwire configuration file.
/etc/tripwire/tw.pol	Encrypted Tripwire policy file.
/etc/tripwire/twcfg.txt	Plain text Tripwire configuration file.
/etc/tripwire/twpol.txt	Plain text Tripwire policy file.
/var/lib/tripwire/report	Holds Tripwire reports.
/var/lib/tripwire	Holds Tripwire databases.

The Tripwire policy file holds rules used to determine what files and programs to monitor and how they are checked. Rules consist of an object and a property mask. An object is either a directory or file and its entry in the rule consists of the full pathname for that file or directory. The property mask is a list of the object's properties to be checked, such as the size, permissions, or a checksum value like MD5. The object and property mask are separated by a **->** symbol, and the entire rule is terminated by a semicolon. You can only have one rule per object. The property mask is a series of single-character codes denoting different file and directory features, such as **p** for permissions, **s** for size, **t** for type, and **M** for MD5 value. You can specify whether a property is to be checked or not with the + and **-** signs. **+p** says to check an object's permissions, **-p** says not to. See the Man page for **twpolicy** for a complete listing of the property codes. In the next example, the **/chris/myfile** object will have its permissions and size checked:

```
/chris/myfile -> +ps;
```

Tripwire also defines several built-in variables that hold standard property sets for different types of objects. For example, ReadOnly lists standard properties for a file or directory that should have read-only access. The Dynamic built-in is used for monitoring files that tend to change. It will check properties like permissions and users that tend not to change, ignoring those like size and MD5 values that do. IgnoreAll will simply check to see if a file exists or not, ignoring all other properties. IgnoreNone will apply all properties to a file. This can be used for providing a high level of security.

```
/usr/bin -> ReadOnly;
/usr/sbin/slogin -> IgnoreNone
/usr/chris/mydoc -> Dynamic
```

You can further qualify rules with attributes such as **severity** to indicate the severity of a violation or **emailto**, in which you can specify an e-mail address to which a message is to be sent in case of a violation. Attributes are entered within parentheses following the rule. Separate several attributes with commas.

```
/chris/myfile -> +ps (emailto = chris@turtle.mytrek.com);
/usr/bin -> ReadOnly (severity = 70, emailto = admin@turtle.mytrek.com);
```

You can also group rules together and apply the same attributes to them all. In this case, the rules are encased in braces and the attributes are listed in preceding parentheses. With this feature, you can avoid having to repeat attributes for several files. Also, you can easily add an attribute for several files at once. In the following example, the **/chris/myfile** and **/chris/myproject** directories are both assigned attributes for an e-mail address and a severity level:

```
 (
 severity = 70,
 emailto = chris@turtle.mytrek.com
 )
 {
 /chris/myfile -> +ps;
 /chris/myproject -> +sM;
 }
```

There are four attributes: **rulename**, **emailto**, **severity**, and **recurse**. The **rulename** attribute is often used to group rules under a title that will then be used in the Tripwire reports to list any violations in that group. The **recurse** attribute specifies if property checks for a directory are also applied to its subdirectories. The default is true, and a false value will not check any files in the directory. In the following example, **rulename** gives the name **Chris Important Files** to the rules listed in the previous example. Be sure to separate attributes with commas. Also the files and subdirectories in the **/chris/myproject** directory are not checked.

```
(
 rulename = "Chris Important Files",
 severity = 70,
 emailto = chris@turtle.mytrek.com
 )
 {
 /chris/myfile -> +ps;
 /chris/myproject -> +sM (recurse = false);
 }
```

Tripwire also supports directives in which you can define variables or rules for certain hosts or file systems, as well as global variables. This allows an administrator to create a single policy file to be used on different hosts (see the **twpolicy** Man page for more details). A directive begins with **@@section**. On a standard Red Hat policy file, you will have a directive for the global variables, GLOBALS, and one for the Linux file system, FS. The GLOBALS section defines locations of Tripwire files and directories. On Red Hat, the FS section sets the variables used for different property sets, such as SEC_INVARIANT that is assigned the properties **+tpug** to check type, permissions, user, and group. This is used for files and directories that should not be changed. You will find the following entry in the Red Hat **twpol.txt** file:

```
SEC_INVARIANT = +tpug ;
```

A variable is evaluated by encasing it with parentheses and preceding it with the **$** operator. $(SEC_INVARIANT) would be use as the property mask in different rules. The following example says that the **/home** directory itself should never be changed (those under it can, **recurse** = 0):

```
/home -> $(SEC_INVARIANT) (recurse = 0) ;
```

On Red Hat, even the built-in variables are also assigned to variables, some with certain qualifications. In the following example, the properties for ReadOnly are assigned to the SEC_BIN variable:

```
SEC_BIN = $(ReadOnly) ; # Binaries that should not change
```

The most widely used variable is SEC_CRIT, which is set to all the properties with IgnoreNone, with the SHa (S) and Havel (H) checksum values and the timestamp (a) property removed:

```
SEC_CRIT = $(IgnoreNone)-SHa ; # Critical files that cannot change
```

Dynamic is used for configuration files, and Growing for log files, as shown here:

```
SEC_CONFIG = $(Dynamic) ; # Config files
SEC_LOG = $(Growing) ; # Files that grow
```

In addition, variables are set for security values. These include SIG_MED, SIG_LOW, and SIG_HIGH for noncritical, moderately critical, and severely critical violations. The following example is a segment of the Red Hat **twpol.txt** file, showing the rules for kernel administrative programs:

```
################################### #
# # #
# Kernel Administration Programs # #
# ##
###################################

(
 rulename = "Kernel Administration Programs",
 severity = $(SIG_HI)
)
{
 /sbin/adjtimex -> $(SEC_CRIT) ;
 /sbin/ctrlaltdel -> $(SEC_CRIT) ;
 /sbin/depmod -> $(SEC_CRIT) ;
 /sbin/insmod -> $(SEC_CRIT) ;
 /sbin/insmod.static -> $(SEC_CRIT) ;
 /sbin/insmod_ksymoops_clean -> $(SEC_CRIT) ;
 /sbin/klogd -> $(SEC_CRIT) ;
 /sbin/ldconfig -> $(SEC_CRIT) ;
 /sbin/minilogd -> $(SEC_CRIT) ;
 /sbin/modinfo -> $(SEC_CRIT) ;
 /sbin/sysctl -> $(SEC_CRIT) ;
}
```

E-mail entries are not included in the Red Hat attributes for different rule groups. If you want Tripwire to notify you by e-mail when a certain violation occurs, you will have to edit the **twpol.txt** file and insert **emailto** attributes into the attribute list for those rule groups. For example, for the previous example you could have Tripwire notify the admin user when a kernel program is violated. Be sure to place a comma at the end of the preceding attribute-in this case, the **severity** attribute:

```
(
 rulename = "Kernel Administration Programs",
 severity = $(SIG_HI),
 emailto=admin@turtle.mytrek.com
)
```

The Tripwire configuration file, **twcfg.txt**, is already set up for a Red Hat Linux installation. It will contain a number of Tripwire variables that you can modify should you wish. The DBFILE variable holds the directory that contains the database file. REPORTFILE specifies the directory where reports are stored. POLFILE contains the policy file. SITEKEYFILE and LOCALKEYFILE specify the location of your local and site key files.

The **twinstall.sh** script will create digitally signed configuration and policy files. To do this, it will prompt you for local and site passphrases. The passphrases are passwords you will need to create a Tripwire database and to access Tripwire reports. You are then prompted to enter the site and local passphrases to generate the configuration and policy files:

```
/etc/tripwire/twinstall.sh
```

twinstall.sh actually runs a **siggen** command to create your passphrases and then the **twadmin** command to create your policy and configuration files. If you later want to change the configuration or policy files, you can run **twadmin** directly, without changing your passphrases. If you want to just change your passphrases, you can run the **siggen** command. To replace both files and both passphrases, you can just run **twinstall.sh** again.

Should you later want to make changes to the Tripwire configuration file, you can edit the **/etc/tripwire/twcfg.txt** file and use it with the **twadmin** command to create a new signed **tw.cfg** file, as shown here:

```
twadmin --create-cfgfile /etc/tripwire/twcfg.txt
```

Creating a new policy file is more complicated and is covered in the policy update section.

Using Tripwire

Once the configuration and policy files have been created, you can create the database of signatures for your monitored files and programs by invoking Tripwire with the **--init** option. You will initially be prompted to enter your local passphrase, which you specified when you ran **twinstall.sh**:

```
tripwire --init
```

The Tripwire datebase is kept in **/var/lib/tripwire** and given the name of the host with the extension **.twd**. For example, the Tripwire database for **turtle.mytrek.com** will be **/var/lib/tripwire/turtle.mytrek.com.twd**.

Note It is recommended, for strong security, to place the Tripwire database on read-only media, like a floppy disk or CD-ROM. You can reconfigure specifying a new Tripwire database directory, or use the **-d** option to manually specify the location of the Tripwire database with your Tripwire commands.

Now that your database is created, you can use Tripwire to periodically check the integrity of your system. You can do this manually with the **--check** option. You could also set up Tripwire **--check** commands as cron jobs to be run automatically at specified times. Red Hat will place a Tripwire **--check** command in the **/etc/cron.daily** file and will run Tripwire daily:

```
tripwire --check
```

Tripwire will check all the files listed in your policy file and generate a report. Tripwire reports are placed in files bearing as their name the hostname, date, and time of the report, with the extension **.twr**. For example, the report generated on August 12, 2001 at 10:29:54 will have the name **20010812-102954.twr**. These files are kept in the **/var/lib/tripwire/report** directory. The report will list any violations, noting a severity level and indicating whether files were added, removed, or modified.

Note If you performed a standard install, you will notice many error messages for missing files. Red Hat assumes a full installation and has set the Tripwire policy file to check for all Red Hat software files. Many of these will be in the Root Config Files rule and in the System Boot Changes rule (daemons). You will have to edit the **twpol.txt** file to remove these entries, and then generate a new policy and database file (see policy update section).

To view reports, you use the **twprint** command with the **--print-report** option. You will have to specify the file you want with the **-r** option as well. The report is displayed on the standard input, scrolling across your screen. You can redirect it to a file to save it, or pipe it to the **more** or **less** commands to view it screen by screen. The following example opens the **20010812-102954.twr** report, piping the output to the **more** command:

```
twprint --print-report -r /var/lib/tripwire/report/20010812-102954.twr |
more
```

You can also use **twprint** to query the database for information about particular files. Use the **--print-dbfile** option and the filename:

```
twprint --print-dbfile /etc/passwd
```

As your system changes with files being modified, your Tripwire database can become outdated. You can update the Tripwire database to incorporate those reported violations as correct entries by using the **--update** option. You will have to specify the particular report file that holds the errors reports for the valid data.

```
tripwire --update -r /var/lib/tripwire/report/20010812-102954.twr
```

Tripwire will first open the file in an editor with those selected as updates to be incorporated having an [x] in front of their entries (unselected ones will have empty brackets, [ ]). Should you notice any valid violations that are marked for update, you can deselect them by removing the preceding x. When finished, save and quit the file with the Editor's **save** command (the EDITOR variable in the **twcfg.txt** file determines what editor to use-Vi by default). You are then prompted to enter a local passphrase for your local key.

Note To control the monitoring of files that no longer exist or are newly installed on your system, you will have to change the policy file (see the next section). The update procedure only deals with modification or feature changes.

Changing Policies

As your system changes over time, you may want to add or remove files that you want to have monitored by Tripwire. The situation becomes aggravated as you install and remove software, adding files not covered by Tripwire and removing those that no longer exist. Tripwire will report any removed files as violations. You can easily add or remove files that Tripwire monitors by inserting or deleting entries in the Tripwire policy file. You can also change the level of checking for different files. You do not edit the Tripwire policy file directly. Instead you edit the text version, **/etc/tripwire/twpol.txt**. If you are removing missing file entries, it is advisable to just comment them out. If you later install the software for them, you will just have to remove the comment. For example, to remove the entries for innd (the INN news server) and for tux (the Tux Web server), just insert a # symbol before their entry as shown here:

```
# /var/lock/subsys/innd -> $(SEC_CONFIG) ;
# /var/lock/subsys/tux -> $(SEC_CONFIG) ;
```

Once you have made your changes, you issue the following command to generate a new signed **tw.pol** policy file that Tripwire will actually use. Be sure to specify the text version you are using.

```
twadmin --create-polfile /etc/tripwire/twpol.txt
```

You will then be prompted to enter the site key. A new **tw.pol** file is then generated. You will then have to regenerate a new version of the Tripwire database. First remove the old one, and then initialize a new one:

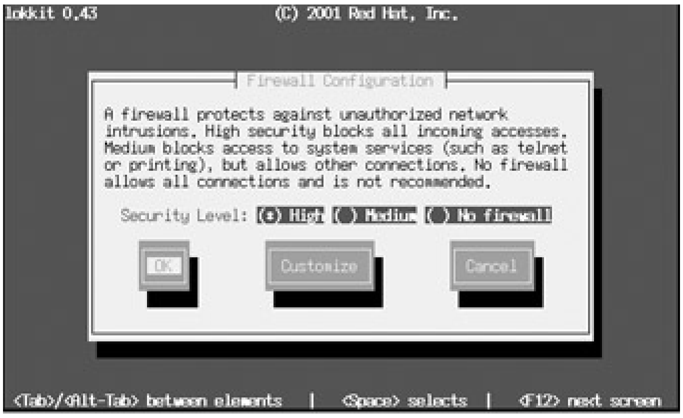
```
rm /var/lib/tripwire/turtle.mytrek.com.twd
tripwire --init
```

Alternatively, you can combine the process by using the **tripwire** command with the **--update-policy** option to create your policy and update your database:

```
tripwire --update-policy /etc/tripwire/twpol.txt
```

Setting Up a Simple Firewall with lokkit

The process of setting up and maintaining a firewall can be complex. To simplify the process, you can use a firewall configuration tool, such as lokkit. lokkit provides a configuration interface for maintaining simple firewall rules. With lokkit, you can select the level of firewall protection as High, Medium, or None (see Figure 6-2). You can run lokkit by entering the command **lokkit** at a command line or in a terminal window. See Chapters 7 and 40 for detailed information on setting up a firewall.

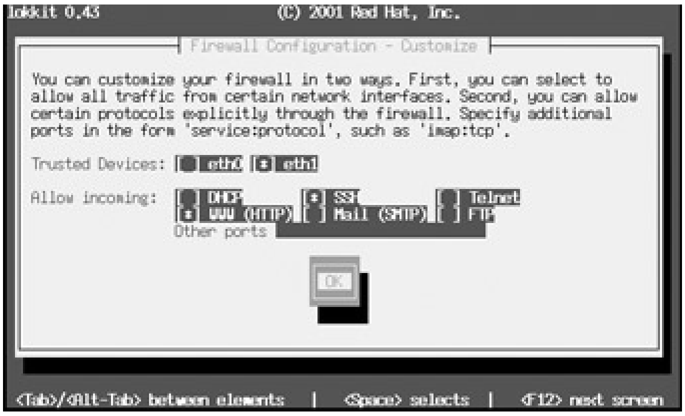


```
lokkit 0.43                    (C) 2001 Red Hat, Inc.

              Firewall Configuration

  A firewall protects against unauthorized network
  intrusions. High security blocks all incoming accesses.
  Medium blocks access to system services (such as telnet
  or printing), but allows other connections. No firewall
  allows all connections and is not recommended.

  Security Level: (*) High ( ) Medium ( ) No firewall

<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next screen
```

Figure 6-2: lokkit

Tip You can also run **gnome-lokkit**, which will prompt you for different settings using Gnome dialog boxes.

An advanced option lets you select different services to be allowed such as mail, the secure shell, FTP, the Web server, and the FTP server (see Figure 6-3). For any service that is not listed, you can enter its name and the protocol it uses manually, such as **imap:tcp** for the IMAP mail service using the TCP protocol.



```
lokkit 0.43                    (C) 2001 Red Hat, Inc.

        ┤ Firewall Configuration - Customize ├

 You can customize your firewall in two ways. First, you can select to
 allow all traffic from certain network interfaces. Second, you can allow
 certain protocols explicitly through the firewall. Specify additional
 ports in the form 'service:protocol', such as 'imap:tcp'.

 Trusted Devices: [ ] eth0 [*] eth1

 Allow incoming:  [ ] DHCP        [*] SSH          [ ] Telnet
                  [*] WWW (HTTP)  [ ] Mail (SMTP)  [ ] FTP
                  Other ports

                              OK

 <Tab>/<Alt-Tab> between elements   |   <Space> selects   |   <F12> next screen
```

Figure 6-3: lokkit advanced options

lokkit uses the older IP Chains firewalling rules and places them in the **/etc/sysconfig/ipchains** file.

Should you want to remove your firewall, you can use lokkit and select the No Firewall option. lokkit is also used by the installation program to set up your firewall, enabling ipchains.

Note lokkit only supports the older IP Chains, not the newer IP Tables. If you want to use IP Tables, you first have to remove the **/etc/sysconfig/ipchains** file and then use the Service Configuration tool, chkconfig, or setup to remove ipchains as a startup service. You can then set up iptables as a startup service and create an **/etc/sysconfig/iptables** file for its rules (see Chapter 40).

If you need more refined firewall protection than lokkit can provide, you can use the packet-filtering program iptables in which you can manually list your firewall rules. With the **iptables** command, you can enter rules with which you can control access to your system. iptables has been developed by the Netfilter Project at **netfilter.samba.org**. The iptables program is the successor to ipchains, used on older versions of Linux. See Chapter 40 for a detailed description of iptables.

Proxies (Squid)

Squid is a proxy-caching server for Web clients, designed to speed Internet access. It implements a proxy-caching service for Web clients that caches Web pages as users make requests and provides security controls for Web site access. Squid is supported and distributed under a GNU Public License by the National Laboratory for Applied Network Research (NLANR) at the University of California, San Diego. The work is based on the Harvest Project. You can obtain current source code versions and online documentation from the Squid home page at **http://squid.nlanr.net** and the Squid FTP site at **ftp.nlanr.net**.

Tip You can also configure Squid using Linuxconf or Webmin. In Webmin, select the Squid Proxy Server from the Servers page.

To configure Squid to provide security to your Web server, you first define access control lists (ACL) using the **acl** command, in which you create a label for the systems on which you are setting controls. You then use commands such as **http_access** to define these controls. You can define a system, or a group of systems, according to several **acl** options, such as the source IP address, the domain name, or even the time and date. For example, the **src** option is used to define a system or group of systems with a certain source address. To define a **mylan acl** entry for systems in a local network with the addresses 192.168.1.0 through 192.168.1.255, use the following ACL definition:

```
acl mylan src 192.168.1.0/255.255.255.0
```

Once these are defined, you can use an ACL definition in a **Squid** option to specify a control you want to place on those systems. For example, to allow access by the mylan group of local systems to the Web through the proxy, use an **http_access** option with the **allow** action specifying **mylan** as the **acl** definition to use, as shown here:

```
http_access allow mylan
```

By defining ACLs and using them in **Squid** options, you can tailor your Web site with the kind of security you want. See Chapter 18 for a more detailed discussion of Squid.

Secure Shell (SSH)

Although a firewall can protect a network from attempts to break in to it from the outside, the problem of securing legitimate communications to the network from outside sources still exists. A particular problem is one of users who want to connect to your network remotely. Such connections could be monitored, and information such as passwords and user IDs used when the user logs into your network could be copied and used later to break in. One solution is to use SSH for remote logins and other kinds of remote connections such as FTP transfers. SSH encrypts any communications between the remote user and a system on your network.

SSH was originally designed to replace remote access operations, such as rlogin, rcp, and telnet (see Chapter 18) as well as FTP. The ssh-clients package contains corresponding SSH clients to replace these applications. With slogin or ssh, you can log in from a remote host to execute commands and run applications, much as you can with rlogin and rsh. With scp, you can copy files between the remote host and a network host, just as with rcp. With sftp you can transfer FTP files secured by encryption.

Unlike PGP, SSH uses public key encryption for the authentication process only. Once authenticated, participants agree on a common cipher to use to encrypt transmission. Authentication will verify the identity of the participants. Each user who intends to use SSH to access a remote account first needs to create the public and private keys along with a passphrase to use for the authentication process. A user then sends his or her public key to the remote account they want to access and installs the public key on that account. When the user attempts to access the remote account, that account can then use the user's public key to authenticate that the user is who he or she claims to be. The process assumes that the remote account has set up its own SSH private and public key. For the user to access the remote account, he or she will have know the remote account's SSH passphrase. SSH is often used in situations where a user has two or more accounts located on different systems and wants to be able to securely access them from each other. In that case the user already has access to each account and can install SSH on each, giving each its own private and public keys along with their passphrases.

You create a SSH public and private keys and select a passphrase, with the **ssh-keygen** command. The **ssh-keygen** command prompts you for a passphrase, which it will use as a kind of password to protect your private key. The passphrase should be several words long. The **ssh-keygen** command generates the public key and places it in your **.ssh/identity.pub** file; it places the private key in the **.ssh/identity** file. If you need to change your passphrase, you can do so with the **ssh-keygen** command and the **-p** option. Each user will have his or her own SSH configuration directory, called **.ssh**, located in their own home directory. The public and private keys, as well as SSH configuration files, are placed here.

Note The **.ssh/identity** file name is used in SSH version 1, still distributed by default in Red Hat 7.2. SSH version 2 uses a different file name, **.ssh/id_dsa**. The authorized keys file is also slightly different, **.ssh/authorized_keys2**.

A public key is used to identify a user and its host. You use the public key on a remote system to allow that user access. The public key is placed in the remote user account's **.ssh/authorized_keys** file. Recall that the public key is held in the **.ssh/identity.pub** file. If a user wants to log in remotely from a local account to an account on a remote system, he or she would first place their public key in the **.ssh/authorized_keys** file in the account on the remote system they wants to access. If the user **larisa** on **turtle.mytrek.com** wants to access the **aleina** account on **rabbit.mytrek.com, larisa**'s public key from **/home/larisa/.ssh/identity.pub** first must be placed in **aleina**'s **authorized_keys** file, **/home/aleina/.ssh/authorized_keys**.

With ssh, you can log in from a local site to a remote host on your network and then send commands to be executed on that host. ssh is also capable of supporting X Window System connections. This feature is automatically enabled if you make an ssh connection from an X Window System environment, such as Gnome or KDE. A connection is set up for you between the local X server and the remote X server. The remote host sets up a dummy X server and sends any X Window System data through it to your local system to be processed by your own local X server.

To log in with SSH, you enter the **ssh** command with the address of the remote host, followed by an **-l** option and the login name (username) of the remote account you are logging into. You will then be prompted for the remote account's SSH passphrase. The following example logs into the **aleina** user account on the **rabbit.mytrek.com** host:

```
ssh rabbit.mytrek.com -l aleina
```

You use **scp** to copy files from one host to another on a network. Designed to replace rcp, scp actually uses ssh to transfer data and employs the same authentication and encryption. If authentication requires it, scp requests a password or passphrase. Directories and files on remote hosts are specified using the username and the host address before the filename or directory. The username specifies the remote user account that scp is accessing, and the host is the remote system where that account is located. You separate the user from the host address with an at sign (**@**), and you separate the host address from the file or directory name with a colon (**:**). The following example copies the file **party** from a user's current directory to the user **aleina**'s **birthday** directory, located on the **rabbit.mytrek.com** host:

```
scp party aleina@rabbit.mytrek.com:/birthday/party
```

Of particular interest is the **-r** option, which enables you to copy whole directories. In the next example, the user copies the entire **reports** directory to the user **justin**'s **projects** directory:

```
scp -r reports justin@rabbit.mytrek.com:/projects
```

Chapter 7: Setting Up a Local Area Network with Red Hat

Overview

Creating a local network of your own involves just a few simple steps. You can set up a Red Hat system to server as the main server for your own local area network (LAN), providing services like e-mail, a Web site, or shared printers. You can even connect different types of systems such as those running Windows or the Mac OS. You can also configure you system to serve as a gateway to the Internet, through which all your other systems will connect. In fact, you could have one Internet connection on your gateway that each host on your network could use. A few security precautions allow your system to work as firewall, protecting your local hosts from outside attacks. You could also set up a very simple configuration to provide Web access only. This chapter will cover the basic procedures for setting up such a network. Later chapters in this book will cover these topics in detail.

Your local area network consists of a collection of host systems connected to the main host running Red Hat Linux. This main host will be referred to as the *gateway*. The steps for setting up a local network involve the following:

- Setting up and configuring the Ethernet cards on each system. Your Red Hat gateway should have two Ethernet cards.
- Setting up a proxy server to provide direct Web access (DNS not required).
- Setting up your DNS server on the Red Hat gateway.
- Configuring your DNS server to allow all other local hosts to access the Internet.
- Setting up firewall protection.
- Enabling e-mail services.
- Setting up local host access to the Internet through DNS (proxy server not required).
- Sharing printers with Windows hosts.
- Setting up a local Web site.

Along with setting up your connections, you will have to run at least one service on the main gateway computer you set up for your network. You can start and stop a service with the **service** command, and have the service automatically started with the Text Mode Setup utility. For a simple network, you should have the DNS and Network services running. If you have Windows systems on your network and you want to share printers with them, you will need the Samba service. The Network, Squid, Sendmail, DNS, and Samba may have to be restarted as you configure them. You will have to know the names used for the DNS, Sendmail, Squid, and Samba server programs to restart them with the service tool. They are shown here. In addition, you will have to add a firewall rule to enable your local hosts to access the Internet through your firewall.

Service Name	Service Program

Service Name	**Service Program**
Domain Name Service (DNS)	named
Samba	smb
Network connections	network
Firewall	iptables
Squid	squid
Sendmail	sendmail

You use the **start**, **stop**, and **restart** arguments to start, stop, and restart a service. To restart the DNS service you would use the following:

```
service named restart
```

To have a service start automatically, select the Text Mode Setup Tool from the Gnome System menu, then use the arrow keys to move down to the System Services entry and press ENTER. This will list the different service programs such as smb and named. To start the DNS service automatically, use the arrow keys to move to the named entry and press the SPACEBAR to select it (some services may already be selected). Use the TAB key to move to the OK button and press ENTER.

Physical Configuration

To set up the physical connections between different computers on your system, you will need to install an Ethernet device on each. Some computers, such as Mac systems, may have this device built-in. Many computers may already have an Ethernet card installed. Most Ethernet configurations use lightweight cables to connect computers, though there are some that are wireless. The computers on a network are referred to as hosts. To connect several hosts together on a network, you will need Ethernet cables for each and a hub that will connect them all together. To connect up a host, connect one end of the cable to its Ethernet card and the other to the hub. A hub will have several plugs, one for each host on your network. For a larger network, you can connect several hubs together.

In the configuration described here, the host running a Red Hat Linux system will be used as the main server and gateway for the local network. Here, you will install various servers like the DNS and Web servers. This host also will function to connect all the local hosts to the Internet (or a larger network). To do this effectively, this gateway/server host will need an Ethernet card and an Internet connection device such as a modem, DSL (digital subscriber line) modem, or another Ethernet card. The type of device you use depends on the type of service that your Internet service provider (ISP) gives you. Some provide only modem connections in which you dial in to connect to the Internet (AOL connections do not work for a LAN). Those that provide DSL connections will use a special DSL modem to allow you connect to the Internet. Both connect to a phone outlet. Cable modems, however, work like Ethernet networks. You need a second Ethernet network card that you connect to the cable modem. This is also the case if you are connecting directly to a larger Ethernet network. The examples in this chapter use a second Ethernet connection.

Note Another kind of network configuration uses a coaxial cable (thin Ethernet cable) to which hosts connect directly instead of to a hub.

When you start up your Red Hat system, Red Hat will automatically detect your Internet connection device and install the appropriate module for it. For some older Ethernet cards, you may have to perform special configuration tasks, such as making entries in **/etc/modules.conf** with certain parameters (see "Ethernet Parameters" in the Red Hat Reference Guide).

Web Access with Squid

If you want only to provide your hosts Internet Web access, you can do so by just running the Squid server on your gateway host. You will not have to set up and run a DNS server. Squid is a proxy server and can handle the Internet connection between a browser and Internet sites directly. You only have to configure the network connections for each host, providing their IP address. Squid is included with the basic installation.

Note Squid also provides extensive security options, making it advisable to control Web access through a proxy server like Squid.

Once it is installed on the gateway host, you then have to configure Squid to allow access by hosts on your network. Edit the **/etc/squid/squid.conf** file and place the following entries in the security section.

```
acl mylan src 192.168.0.0/255.255.255.0
http_access allow mylan
```

The squid.conf file is a very large file with default settings commented in detail. An easy way to make your entries is to search for the corresponding localhost entries and add your network ones below them. The acl entry for localhost will begin with "acl localhost". The one for access will begin with "http_access allow localhost". Squid configuration is discussed in detail in Chapter 27. You can also use Linuxconf or Webmin to configure Squid.

Once it is configured, you can run Squid with the **service** command.

```
service squid start
```

Use **chkconfig** to have it start automatically when you boot.

```
chkconfig --level 35  squid on
```

When configuring a Web browser, select the Proxy option and enter for the proxy server, the IP address of the gateway running Squid and port 3128. On Netscape select the Proxy entry under Advanced in the Options panel, and then view the manual proxy connections. For example, using the sample network described in this chapter, the Squid proxy server would be running on 192.168.0.1 and use port 3128. So the entry used in Web browsers would be the following for the different servers:

```
192.168.0.1
```

And then use 3128 for the port:

```
3128
```

Now any user on your network with a correctly configured browser can access the Web.

DNS Setup

Now that your local network is physically set up, your Red Hat gateway/server needs to run certain services to allow your hosts to communicate over the network. You first have to configure and run a Domain Name Service (DNS), which will allow all the hosts on your local network to identify each other using a hostname. This involves several steps:

1. Decide on the IP addresses to assign to each local host. Use 192.168.1 as the network address.
2. Decide on the domain name for your local network.
3. Decide on the hostname for each host on your network.
4. Each host has to be configured with its IP address and domain name address.
5. On the Red Hat gateway/server, configure a DNS server listing each host's IP address and hostname.
6. Start the DNS service.

All hosts on the Internet are identified by their IP addresses. When you send a message to a host on the Internet, you must provide its IP address. Using a sequence of four numbers of an IP address, however, can be difficult. They are hard to remember, and it's easy to make mistakes when typing them. To make identifying a computer on the Internet easier, the Domain Name Service (DNS) was implemented. The DNS establishes a fully qualified domain name address for each IP address. The fully qualified domain name consists of the name of the host and the network (domain) that it belongs to. Whenever you use that name, it is automatically converted to an IP address, which is then used to identify that Internet host. The fully qualified domain name is far easier to use than its corresponding IP address. For example, the name **www.linux.org** has an IP address of 198.182.196.56. A DNS server will translate **www.linux.org** into its IP address, 198.182.196.56.

In Figure 7-1 the user at **rabbit.mytrek.com** wants to connect to the remote host **lizard.mytrek.com**. **rabbit.mytrek.com** first sends a request to the network's DNS server-in this case, **turtle.mytrek.com**-to look up the name **lizard.mytrek.com** and find its IP address. It then returns the IP address for **lizard.mytrek.com**, 192.168.0.3, to the requesting host, **rabbit.mytrek.com**. With the IP address, the user at **rabbit.mytrek.com** can then connect to **lizard.mytrek.com**.

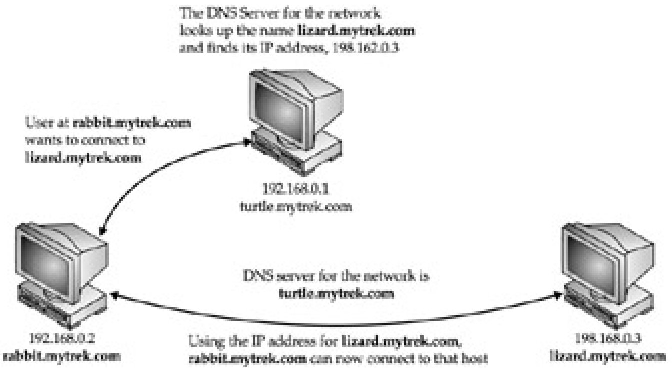

The DNS Server for the network
looks up the name lizard.mytrek.com
and finds its IP address, 198.162.0.3
User at rabbit.mytrek.com
wants to connect to
lizard.mytrek.com
192.168.0.1
turtle.mytrek.com
DNS server for the network is
turtle.mytrek.com
192.168.0.2
rabbit.mytrek.com
Using the IP address for lizard.mytrek.com,
rabbit.mytrek.com can now connect to that host
198.168.0.3
lizard.mytrek.com

Figure 7-1: DNS server operation

You can then set up domain name services for your network by running a DNS server on one of the machines. This machine becomes your network's DNS server. You can then give your machines fully qualified domain names and configure your DNS server to translate the names to their corresponding IP addresses. As shown in Figure 7-2, for example, you could give the machine 192.168.0.1 the name **turtle.mytrek.com**, and the machine 192.168.0.2 the name **rabbit.mytrek.com**. You can also implement Internet services on your network such as FTP, Web, and mail services by setting up servers for them on your machines. You can then configure your DNS server to let users access those services using fully qualified domain names. For example, for **mytrek.com** network, the Web server could be accessed using the name **www.mytrek.com**.

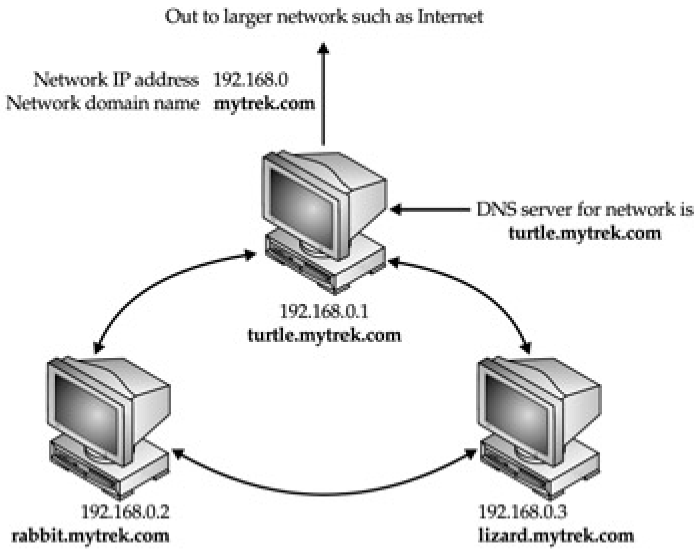

Out to larger network such as Internet
Network IP address 192.168.0
Network domain name mytrek.com
DNS server for network is turtle.mytrek.com
192.168.0.1
turtle.mytrek.com
192.168.0.2
rabbit.mytrek.com
192.168.0.3
lizard.mytrek.com

Figure 7-2: DNS server and network

Note Instead of a Domain Name Service, you could have the **/etc/hosts** files in each machine contain the entire list of IP addresses and domain names for all the machines in your network. But, for any changes, you would have to update each machine's **/etc/hosts** file.

IP Addresses

Most networks, including the Internet, use a set of network protocols called TCP/IP, which stands for Transmission Control Protocol/Internet Protocol. On a TCP/IP network such as the Internet, each computer is given a unique address called an *IP address.* The IP address is used to identify and locate a particular host-a computer connected to the network. It consists of a number, usually four sets of three numbers separated by periods. An example of an IP address is 192.168.0.1.

You will have to assign an IP address to each host on your network. The IP address consists of a number composed of four segments separated by periods. Depending on the type of network, several of the first segments are used for the network address and several of the last segments are used for the host address. For a small local network, the first three segments are the computer's network address and the last segment is the computer's host ID (as used in these examples). For example, in the address 192.168.0.2, 192.168.0 is the network address and 2 is the computer's host ID within that network. Together, they make up an IP address with which the computer can be addressed from anywhere on the Internet. IP addresses, though, are difficult to remember and easy to get wrong.

To set up a DNS server for a local area network (LAN) whose hosts are not directly connected to the Internet, you would use a special set of IP numbers reserved for such non-Internet networks (also known as *private networks* or *intranets*). This is especially true if you are implementing IP masquerading, where only a gateway machine has an Internet address, and the others make use of that one address to connect to the Internet. For a small network (254 hosts or less), these are numbers that have the special network number 192.168.0, as used in these examples. If you are setting up a LAN, such as a small business or home network, you are free to use these numbers for your local machines. For a local network, assign IP addresses starting from 192.168.0.1. The host segment can range from 1 to 254, where 255 is used for the broadcast address. If you have three hosts on your home network, you can give them the addresses 192.168.0.1, 192.168.0.2, and 192.168.0.3.

The network address for such a network would be the first three segments of the IP address, 192.168.0. The network netmask would cover those first three segments, using the number 255.255.255.0. The network netmask is used to determine the host and network parts of an IP address. The broadcast address is used to allow an administrator to contact all hosts at once. You would then use these three IP addresses when configuring a host.

Network IP address	192.168.0.0
Network netmask	255.255.255.0
Host IP addresses	From 192.168.0.1 to 192.168.0.254
Broadcast address	192.168.0.255

In the sample network used in these examples, there are three hosts, each with its own IP addresses and hostnames listed here. The network address, netmask, and broadcast address are the same as those listed above:

```
192.168.0.1
192.168.0.2
192.168.0.3
```

Figure 7-3 shows the format of the sample network with their Ethernet connections and IP addresses, along with their hostnames.

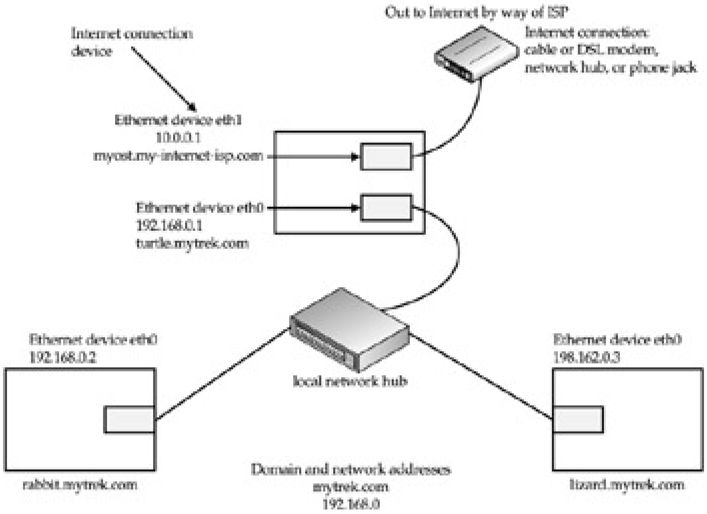


Figure 7-3: Sample local network Ethernet connections, IP addresses, and hostnames

IP Addresses for the Gateway

Though it may look like a single IP address is assigned to an entire computer, it is important to realize that this is not actually the case. An IP address is really assigned to a network device such as an Ethernet card or DSL modem. Ordinary computers on a network will have only one network device, giving them only one IP address. However, a gateway computer will normally have at least two network devices, each with its own IP address. The device used for the local network will have the IP address you decided to give it. The device used for the Internet connection will have an IP address assigned to it by your ISP. For example, in the sample network used in these examples, the computer used as the gateway will have two Ethernet cards, one for the local network and one for the Internet. Each will have its own IP address. The Ethernet card used for the local network will have the address 192.168.0.1, and the one used for the Internet connection will have the address 10.0.0.1 assigned by the ISP (a fabricated address used for this example):

```
10.0.0.1
```

Along with its IP address, each device will have its own hostname, as described in the next section. The Ethernet card for the local network will have the hostname you decided to give it, and the Ethernet card connected to the Internet will have a hostname assigned to it by your ISP. The hostnames used in these examples are **turtle.mytrek.com** for the local Ethernet card and **myhost.my-internet-isp.com**.

The Internet connection device on the gateway will either use a static or dynamic IP address. A static IP address, such as those used for cable and DSL modems, will remain the same. This is the IP address you would assign to your Internet network device. The sample network described here uses a static address, 10.0.0.1. If your Internet connection device is a modem (in some cases, also DSL), your IP address is dynamic. Your ISP assigns you a different one each time you connect from a pool it keeps on hand. Since your Internet IP address keeps changing, you do not know what it will be any given time you connect. For this situation, when you have to reference the Internet IP address in your configurations, you reference the Internet network device instead.

Domain Name

Next you will have to decide on a domain name for your local network. The domain name is the name used to identify your network. It will be translated into the network part of the IP address, the first three segments. The domain name can be any name you want to give it. The extension is used to denote the type of domain:

```
domain-name.extension
```

The following is the domain name for a local network called **mytrek**:

```
mytrek.com
```

Hostnames will be attached to the front of the domain name to provide a complete domain name address for a particular host computer. This is referred to as the Fully Qualified Domain Name (FQDN), but actually references a particular host.

Hostnames

For the hosts on your local network, you need to create your own hostnames. The hostname itself can be any name you choose. The term "hostname" is also used to refer to the fully qualified domain name, also referred to as the full hostname. This consists of the hostname attached to the domain name. On a large network such as the Internet, the host is referenced with its fully qualified domain name. The full hostname consists of the hostname, the name you gave to your computer; a domain name, the name that identifies your network; and an extension that identifies the type of network you are on. Here is the syntax for the fully qualified domain name:

```
host-name.domain-name.extension
```

In the following example, the fully qualified domain name references a computer called **turtle** on a network referred to as **mytrek**. It is part of a commercial venture, as indicated by the extension **com**:

```
turtle.mytrek.com
```

For hosts within a local network, hosts can reference each other using just their hostname, without the domain name or extension:

```
turtle
```

Note For hosts connected directly to the Internet, their domain name and IP addresses are officially registered with an Internet domain name registry like the American Registry for Internet Number (ARIN) so that each computer on the Internet can have a unique name. This is handled by your ISP, who will then give your computer a unique hostname with the ISP's domain name.

Configuring Hosts

For each host on your network, you will have to enter network configuration information such as the IP address and hostname that you decided to give this host. Other information like the netmask and the host that will run the local network's DNS server will also be needed.

The sample network described here uses three hosts in the domain **mytrek.com** with the IP addresses shown here:

```
turtle.mytrek.com 192.168.0.1
rabbit.mytrek.com 192.168.0.1
lizard.mytrek.com 192.168.0.1
```

In addition, the gateway, **turtle.mytrek.com**, will have another Ethernet card that functions as the Internet connection device. It will have the hostname and IP address shown here:

```
myhost.my-internet-isp.com 10.0.0.1
```

Linux Hosts

To configure any Linux hosts on your network, you just follow the network configuration instructions in Chapter 5. For Red Hat systems, you can use netcfg, the network configuration tool. Be sure to specify that the system holding your Internet connection is specified as your gateway. For example, the **rabbit.mytrek.com** host has to have its DNS name and its IP address (192.168.0.2) entered in its **/etc/hosts** file. You do this on netcfg using the Hosts panel and entering the DNS name and the IP address. You also have to specify the interface used and its IP address. On netcfg, you do this by clicking on the Interface panel and entering the Ethernet card device name (**eth0**) and its IP address. The gateway it will use is **turtle.mytrek.com** at 192.168.0.1. On netcfg, you do this by clicking on the Routing panel and entering the IP address of the gateway in the Default Gateway box. The gateway computer will have a slightly different configuration in that you will also have to add Internet connection information. With netcfg, on the Names panel you add the DNS name and IP address given by your ISP for your host. On the Interface panel, add the network device used for the Internet connection, giving its device name and IP address assigned by the ISP. In this example, it is the second Ethernet card, **eth1**, with the IP address 10.0.0.1. If you are using a dial-up modem and your ISP provides you with a dynamic IP address, you simply specify the Internet device, such as **ppp0**, instead of the IP address. On the Routing panel, you enter the gateway used by the ISP in the Default Gateway box-in this example, it is 10.0.24.1. Do not use the gateway for your local network. Also, be sure to click the check box labeled Network Packet Forwarding to enable your local hosts to access the Internet.

The DNS servers also have to be specified in the **/etc/resolv.conf** file. You can do this with netcfg on the Names panel. In this example, the DNS server for the local network hosts would be 192.168.0.1, **turtle.mytrek.com**. The gateway host with the Internet connection would also have any DNS servers provided by the ISP. So the **/etc/resolve.conf** file on **turtle.mytrek.com** will list itself (192.168.0.1), the DNS server for the local network, and any DNS servers provided by the ISP it is connected to. You will see later that the local host can connect through the gateway host to access the Internet and use the ISP DNS servers as if they were the gateway host.

Note You can also list the ISP DNS servers on your local hosts.

Windows Hosts

To configure a Windows host, you need to access the TCP/IP panel that controls its Ethernet card. On Windows 95/98/ME, double-click on the Network icon in the Control Panel window. If there is already a TCP/IP entry for your Ethernet card, click on it. If not, you need to add one by selecting the Ethernet card entry and clicking on the Add button. Click on Protocol to open the Select Network Protocol window, then click Microsoft and double-click TCP/IP. A TCP/IP entry for the Ethernet card will appear.

Double-clicking on the TCP/IP entry opens a TCP/IP Properties window with several panels. Click on IP Address and enter the address you decided to give this host, along with the network mask, 255.255.255.0. Click the DNS Configuration tab and enter the IP address of the DNS server-in this example, it is 192.168.0.1. Click the Gateway tab and enter the IP address for the Linux gateway host-in this example, it is 192.168.0.1.

On Windows 2000, NT, and XP, you select Network and Dialup connections from the Settings menu. This folder will list network connection icons for all of your network connection devices such as Ethernet cards or modems. A local area connection icon will be listed for the Ethernet card connected to your local network. Double-click on it to open a status window, and then click on the Properties button to open the Local Area Connection

Properties window. Click on Internet Protocols in the Components list and click the Properties button. In the Internet Properties window, you can enter the IP addresses for the host address, netmask (subnet mask), default gateway, and DNS server. In the sample network, assuming that **lizard** is a Windows system, you would enter **192.168.0.3** for the IP address, **255.255.255.0** for the netmask, **192.168.0.1** for the default gateway, and **192.168.0.1** for the DNS server.

To set the computer host and domain name, click on System in the Control Panels and select the Network Identification panel. Click the Properties button to enter the hostname and the domain name.

Checking IP Configurations

Once you have set up your connections and configured all your hosts, you can see if the physical and host configurations are working by trying to contact them with the **ping** command. From a host computer, use the **ping** command with the IP address of other hosts to see if they can be accessed. If an IP address is not found, that host is either not connected, its connection device is not working, or its network configuration is faulty. If the connection device is not working, it may require a different driver or certain parameters specified.

```
ping 192.168.0.2
```

If you need to make changes, you can just restart the network service to have them take effect.

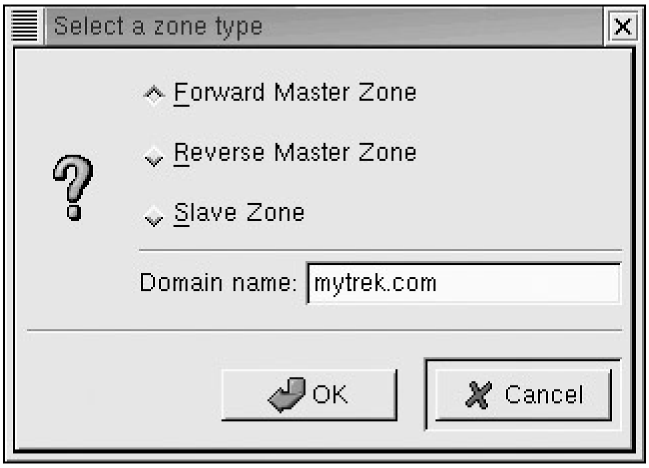
```
service network restart
```

Configuring the DNS Server

To configure the DNS server, you will have to enter the IP addresses and hostnames for your different hosts in the DNS configuration file. You can do so easily with the Bind Configuration tool. Bind is the kind of DNS software used on most networks. Select bindconf from the Gnome System menu. You will need to create two zone configurations: a forward master zone and a reverse master zone.

Note Any computer on the Internet can maintain a file that manually associates IP addresses with domain names. On Linux systems, this file is called the **/etc/hosts** file. Here, you can enter the IP addresses and domain names of computers you commonly access. Using this method, however, each computer needs a complete listing of all other computers on the Internet, and that listing must be updated constantly.

Your forward master zone is where you enter your main DNS configuration entries for the host domain names and their IP addresses. Click the Add button and select Forward Master Zone. You will initially be asked to enter a domain name. Enter the domain name you decided on for your local network, such as **mytrek.com**, as shown here:



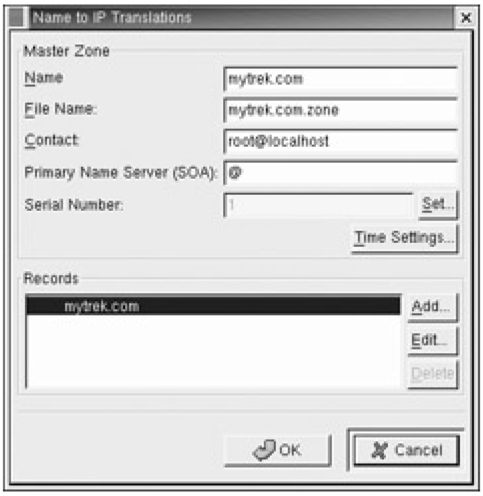
● Forward Master Zone

○ Reverse Master Zone

○ Slave Zone

Domain name: mytrek.com

A window then opens labeled Master Zone with several entries for your DNS server (see Figure 7-4). Default settings based on the domain name you previously entered have already been entered for you. The Name box will hold the domain name you just specified. File_Name is the name for the master zone file and will be the name of the domain along with the extension **.zone**. The e-mail address is the address of the person managing the DNS server, by default set to the root user. Feel free to change this to another user's address. The Primary Name Server entry will have an **@** as its default. The **@** symbol merely represents the name of the DNS server. The Serial Number field will have an initial value of 1. It will increment automatically whenever you make changes to the configuration files. The Time Settings button opens a dialog box where you can set refresh, retry, and expiration dates. You can leave these as they are. These, as well as the other settings, are covered in detail in Chapter 25.



Master Zone

Name: mytrek.com

File Name: mytrek.com.zone

Contact: root@localhost

Primary Name Server (SOA): @

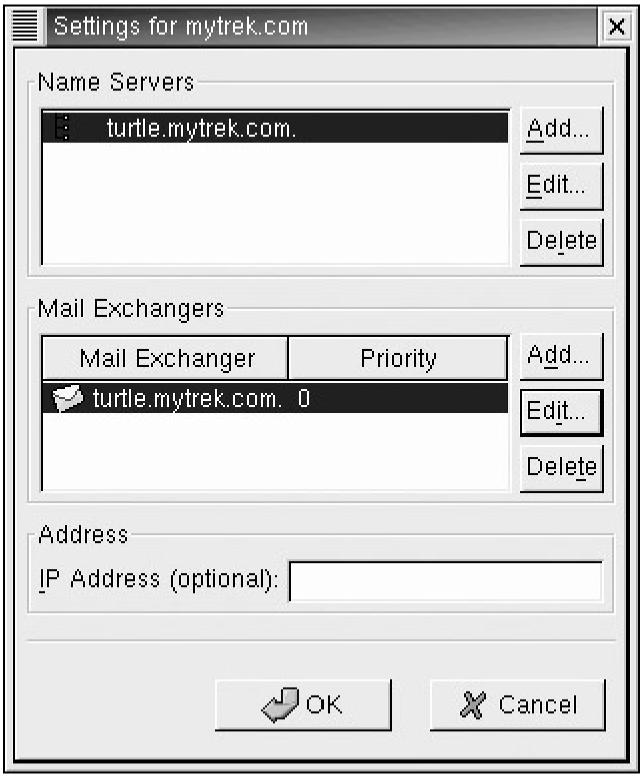
Serial Number: 1

Records

mytrek.com

Figure 7-4: bindconf Master Zone screen

In the Records section of the Master Zone window, you add and edit the host and domain entries. A domain entry will already be added for you. It will bear the name of your domain. You will have to edit this entry to specify the host running the DNS server. If your network has a mail server, as discussed in Chapter 26, then you would enter that here also. The host running the DNS server is referred to as the name server. Usually these are the same. Select the entry and click the Edit button. This opens a window with two sections, one for the name servers and one for the mail servers (larger networks could have several mail servers and slave name servers). Figure 7-5 shows the domain window with two entries added for the name server and the mail server. For a small network, you would not normally have a mail server. Each host would operate as its own mail server and does not need to be specified in the DNS configuration. Larger networks that have centralized mail service operations would make use of DNS mail entries to configure mail delivery on their network (see Chapter 26 for more details).



Settings for mytrek.com

Name Servers

turtle.mytrek.com.

Mail Exchangers

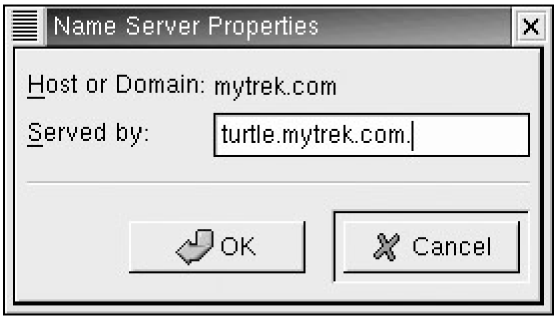
Mail Exchanger	Priority
turtle.mytrek.com.	0

Address

IP Address (optional):

Figure 7-5: Domain window

Click the Add button in the name server section to open a window where you enter the hostname of the name server, as shown next. Be sure to add a period at the end of the name server's hostname. In this example, it is **turtle.mytrek.com**.

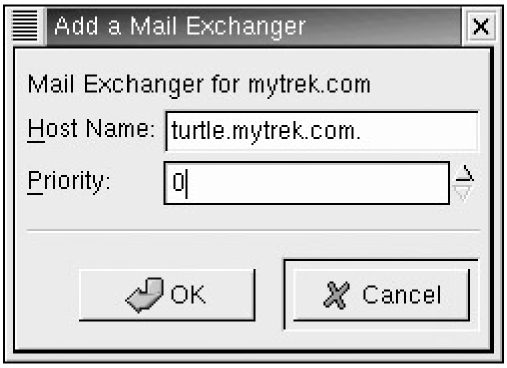


Name Server Properties

Host or Domain: mytrek.com

Served by: turtle.mytrek.com.

Should you have a network with its own centralized mail server, like large networks may have, you can add it now. A small network like a home network would not normally have such a server. Click the Add button in the mail server section to open a window where you enter both the host running the mail server and the priority of the mail server. For a single mail server, just set the priority to 0, as shown here:



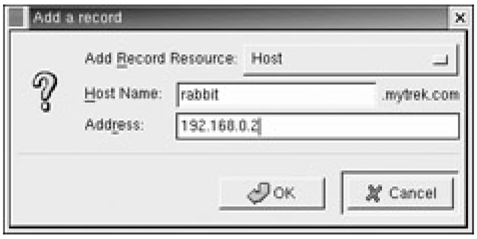
Mail Exchanger for mytrek.com

Host Name: turtle.mytrek.com.

Priority: 0

Once you have added the name and any mail servers, click OK to return to the Master Zone window.

You now need to add entries for the different hosts on your system, providing both their hostname and IP address. To add an entry, you click the Add button. This opens a window with boxes for entering the hostname and its IP address, as shown next. For the hostname, you do not have to include the domain name. That will be automatically added for you. You will see the domain name listed next to the entry box. The host **rabbit** is added for the **mytrek.com** domain with the IP address 192.168.0.2.



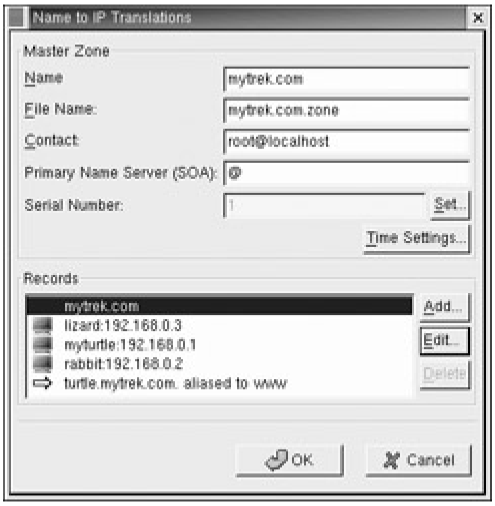
Add Record Resource: Host

Host Name: rabbit .mytrek.com

Address: 192.168.0.2

Note If you want to set up a Web site on the gateway host, you should add an alias for it, where the alias uses the host name **www**. When adding the alias, select Alias from the pop-up menu labeled Add Resource Record.

A completed example of a master zone configuration is shown in Figure 7-6. The domain is **mytrek.com** and there are three hosts: **turtle**, **rabbit**, and **lizard**. Press OK to finish your master zone configuration.



Master Zone

Name: mytrek.com

File Name: mytrek.com.zone

Contact: root@localhost

Primary Name Server (SOA): @

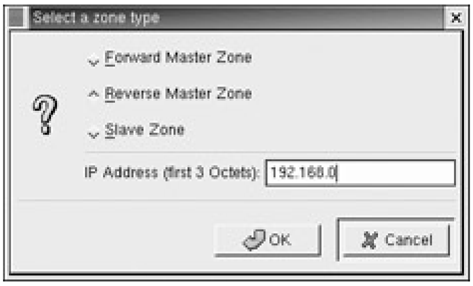
Serial Number: 1

Records

mytrek.com
lizard:192.168.0.3
myturtle:192.168.0.1
rabbit:192.168.0.2
turtle.mytrek.com. aliased to www

Figure 7-6: Master zone example

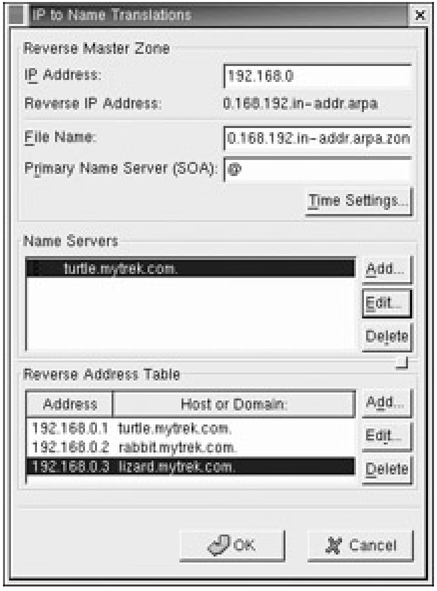
You then have to create a reverse master zone. Click the Add button and select Reverse Master Zone (see the next illustration). In the Zone Type window that first appears, click on the Reverse Master Zone check box. The entry box on this window will then be labeled IP Address (first three octets). Here, enter the network part of your host IP address. This is the first three numbers for the IP addresses you are using for the hosts on your system. For example, the IP address for **turtle.mytrek.com** is 192.168.0.1, so the network part is 192.168.0. The network part will be the same for all your hosts.



IP Address (first 3 Octets): 192.168.0

A Reverse Master Zone window will then open up with the IP address and the File Name already filled in for you (see Figure 7-7). You then need to add entries for all the hosts on your network. In the Reverse Address Table section, click the Add button to open a window where you enter the IP address and hostname for the host. In the IP Address box, enter just the host part of the host's IP address. This will be a single number, usually starting from 1. For example, for the IP address 192.168.0.2, you would just enter 2. In the Full Host Name box, you enter the full hostname of the host, including an ending period. For the rabbit host you would enter

```
rabbit.mytrek.com.
```



IP Address: 192.168.0

Reverse IP Address: 0.168.192.in-addr.arpa

File Name: 0.168.192.in-addr.arpa.zon

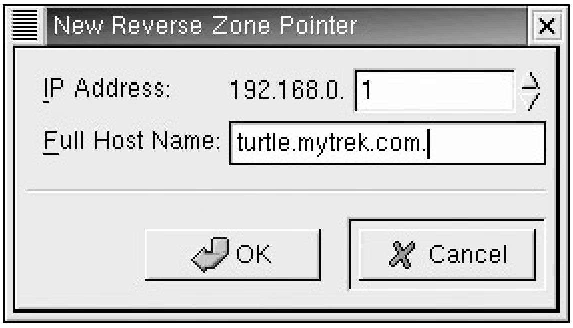
Primary Name Server (SOA): @

turtle.mytrek.com.

Address	Host or Domain:
192.168.0.1	turtle.mytrek.com.
192.168.0.2	rabbit.mytrek.com.
192.168.0.3	lizard.mytrek.com.

Figure 7-7: Reverse master zone

In the next illustration, the information for the **turtle.mytrek.com** host is entered. For the IP address, 1 is added, giving an IP address of 192.168.0.1. For the full hostname, the domain name and an ending period are entered, **turtle.mytrek.com**.



IP Address: 192.168.0. 1
Full Host Name: turtle.mytrek.com.

You then need to add an entry for the DNS server. In the Name Server section click the Add button. Then enter the name of the host that will run your DNS server. Be sure to include a trailing period. In this example, the name server host is

```
turtle.mytrek.com.
```

A completed example of a reverse master zone is shown in Figure 7-7.

To save your bindconf configuration, select the Apply entry to generate the DNS server configuration files. You can then quit.

Starting the DNS Service

To manually start your DNS service, use the **service** command and the service name **named** with the **start** option. **named** is the name of the DNS server used on Linux:

```
service named start
```

When checking for errors or making changes, you can restart the DNS service with the **restart** option:

```
service named restart
```

You can check to see if your DNS server is working by trying to access a particular host using its hostname. Use the **ping** command with the hostname to see if you can make contact. The following command checks to see if the **rabbit** host is running:

```
ping rabbit
```

You can also add the domain name:

```
ping turtle.mytrek.com
```

Logged into the gateway, you can ping your Internet connection to see if it is working:

```
ping myhost.my-internet-isp.com
```

Once everything is working, you can have the DNS service started automatically whenever you start your system. Use the Text Mode Setup Text tool on the Gnome System menu and select System Services. Then, move to the **named** entry and press the SPACEBAR. Tab to the

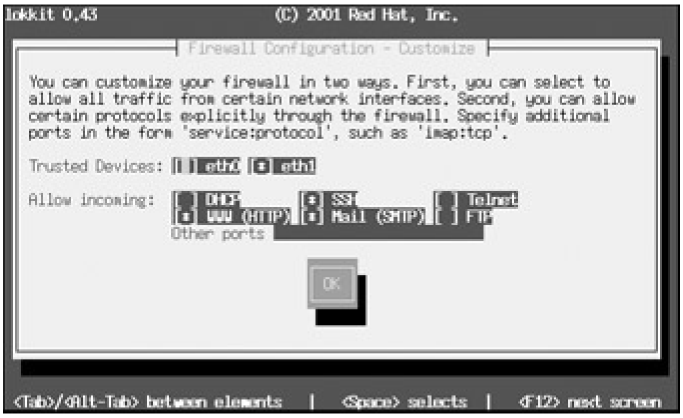
Quit button and press ENTER. Alternatively, you can use **chkconfig** with the **--level** 35 and **on** options:

```
chkconfig --level 35 named on
```

Setting Up Your Firewall

To set up your firewall, run lokkit on the gateway host, as described in Chapter 6. However, be sure that lokkit has not been run on any of the other local hosts, or that any of the local hosts have any kind of firewall running. You can use lokkit to remove the firewall, if necessary. The firewall should only run on the gateway. Furthermore, the gateway will have two network connections, one for the local network and an Internet connection device for the Internet. Make sure that the firewall is applied to the device used for the Internet device, not for your local network. On lokkit you do this by making the local network device a trusted device. Select the Customize button and then, in the list of trusted devices, TAB to the device used for your local network and press the SPACEBAR. An x will appear. Make sure that the firewall network device is left blank.

In the network example used here, the firewall is run on the **eth0** network device (the first Ethernet card), which functions as the gateway. The local network is connected through the **eth1** network device (the second Ethernet card). In Figure 7-8, the **eth1** device is trusted and the eth0 device is not, making it the firewall device.



```
lokkit 0.43                    (C) 2001 Red Hat, Inc.

        Firewall Configuration - Customize

 You can customize your firewall in two ways. First, you can select to
 allow all traffic from certain network interfaces. Second, you can allow
 certain protocols explicitly through the firewall. Specify additional
 ports in the form 'service:protocol', such as 'imap:tcp'.

 Trusted Devices: [ ] eth0 [*] eth1

 Allow incoming:  [ ] DHCP        [*] SSH          [ ] Telnet
                  [*] WWW (HTTP)  [*] Mail (SMTP)  [ ] FTP
                  Other ports

                              OK

<Tab>/<Alt-Tab> between elements  |  <Space> selects  |  <F12> next screen
```

Figure 7-8: lokkit trusted devices

Note lokkit uses the older IP-Chains firewall rules. If you want to use the newer IP Tables rules, you will need to manually enter rules and install them on your system. See Chapter 40 for more details.

If you are creating a strong firewall, but still want to run a service like a Web server, allow users to perform FTP file transfers on the Internet, or allow remote encrypted connections such as SSH, you will have to specify them in the lokkit customization window. Figure 7-8 shows Web, FTP, SSH, and mail communications permitted.

Setting Up E-mail Services

There are several ways to enable e-mail services on your network. You can set up your network in one of two ways: have a central server that handles e-mail for all the users on your network, or have each host handle its own user's independently. An independent setup is the easier and is described here. Central servers are examined in Chapter 26. Internet mail setup also varies depending on whether you have a stand-alone system, a small network with one connection, or a larger network with its own official domain address (see Chapter 26). All setups entail configuring the Sendmail server. Sendmail is an e-mail server used to send and deliver mail on a network. Sendmail is provided with your Red Hat distribution.

Configuring Sendmail involves manually editing the **/etc/mail/sendmail.mc** file and making changes. This is a text file that can be edited using any text editor like emacs, vi, or any of the Gnome or KDE text editors. Be sure to make a backup copy of the file first. Once you have made your changes and saved your file, you then have to install them in Sendmail with the following m4 operation to generate a **/etc/sendmail.cf** file. The **/etc/sendmail.cf** file is the actual Sendmail configuration file.

```
m4  /etc/mail/sendmail.mc > /etc/sendmail.cf
```

You then need to restart Sendmail to have the changes take effect.

```
service sendmail restart
```

Local Network Connections

For messages sent between hosts on your network, you only need to run the Sendmail server on each, making a few changes to their Sendmail configurations. The Sendmail server on one of your hosts can be configured to handle the task of relaying messages between hosts. Using the network example described earlier, the hosts turtle, rabbit, and lizard will be running their own Sendmail servers. The Sendmail server on the turtle host will be configured to relay messages between all the hosts, itself included.

On each host on your network, edit the **/etc/mail/sendmail.mc** file and make the following change. Comment out the **DAEMON_OPTIONS** line in the default Red Hat **sendmail.mc** file by placing a **dnl** word in front of it, as shown here. Removing this feature will allow you to receive messages over your local network. This entry is restricting Sendmail to the localhost (127.0.0.1):

```
dnl DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
```

In the **sendmail.mc** file located on the host that you want to have handle the relaying of messages, you need to also add the following line.

```
FEATURE(relay_entire_domain)dnl
```

Run the m4 operation to install the changed configuration and then restart the server with the service operation, as described earlier.

You can now e-mail messages from one user to another across your network. For example, **george@turtle.mytrek.com** can now email a message to **larisa@rabbit.mytrek.com**. The local Sendmail servers will take care of sending and delivering mail to users both within their hosts and those located on other network hosts.

Internet Connections

To send mail to and from the Internet, you should make use of the e-mail services provided by your ISP. This can vary depending upon the kind of service you have. For a small network, like a home network, you may have only one Internet connection. In this case, your ISP normally provides e-mail services for you such as a number of mailboxes for designated users. If you have a larger network, your ISP may have set up a separate official domain for you and is relaying mail to your network. In this case you can set up your own mail server to handle Internet mail. This is discussed in Chapter 26.

To make use of the mail services that your ISP provides, you can use a mail client like those described in Chapter 17. For a basic Internet connection, ISPs will normally provide you with an e-mail address, along with the choice of several other addresses. You could assign these to different users on your network. Your ISP will be operating an smtp and POP or IMAP mail server to handle mail for their users. Your users could then access their mail on these servers directly. Mail clients like Netscape, Mozilla, and Kmail let you specify a remote smtp or POP server. The smtp server is used to send mail out, and the POP and IMAP servers are used to receive mail. As also discussed in Chapter 17, you can also arrange to have mail delivered directly to a user account on one of your hosts, using fetchmail.

Using your own Sendmail servers to handle Internet mail is a much more complicated process and is described in detail in Chapter 26.

Internet Access by Local Hosts

You can configure your network so that hosts on your local network can access the Internet, using your gateway host's Internet connection. In this scheme, the host will pretend to be the gateway host, using its Internet connection as if it were its own. This way, you only need one Internet connection for all the hosts on your network. The method is called IP masquerading, and it works by the local hosts pretending to have the IP address of the gateway. In effect, all your local hosts share the same Internet IP address. First, make sure that iptables is enabled on your system. You can use the System services list in the Text Mode Setup utility to select iptables for automatic startup.

Tip If you only want to provide Web access to users on your network, you just need to configure and run the Squid proxy server on your gateway. You do not need DNS or IP masquerading implemented.

Note Be sure that a firewall program is not also running on any of your local hosts. This can happen if you ran lokkit on any of your local hosts. Run lokkit again on the local host and select No Firewall. This will shut off the firewall on that host.

IP masquerading is implemented as part of the IP-Chains or IP-Tables firewall programs, depending on the one you are using. If you set up your firewall with lokkit, you are using IP-Chains. To implement IP masquerading, you need to add a new rule to the collection of rules already set up by lokkit. To do so, you need only enter a simple rule on the command line using the **ipchains** command, as shown here. The **-o** option is used to specify the device you are using for your Internet connection. The first Ethernet device, **eth0**, is used in this example. If you are using a modem, the device would be **ppp0** for the first modem device. IP masquerading with iptables is described in detail in Chapter 40.

```
ipchains -A forward -i eth0 -j MASQ
```

You then use the iptables **service** script with the **save** option to save your new rule, along with the ones already set up by lokkit:

```
service ipchains save
```

The rules will be placed in the **/etc/sysconfig/ipchains** file, which will be read whenever you start up your system. Bear in mind that if you run lokkit again, it will overwrite this file. You will have to enter the iptables masquerading command again and save your rules to enable IP masquerading.

Also, check to see if IP forwarding is turned on. You can do this with netcfg on the gateway host. Click the Routing panel and then click the check box at the top labeled Network Packet Forwarding. This sets the value of the **/proc/sys/net/ipv4/ip_forward** file to 1, turning on IP forwarding. You can do this manually instead if you want with the following command.

```
echo  1 > /proc/sys/net/ipv4/ip_forward
```

If you are using IP Tables, you follow much the same procedure, except that you do not use lokkit and you use **iptables** commands to implement firewall rules. The rule to add masquerading with IP Tables for a first Ethernet device (eth0) is as follows:

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

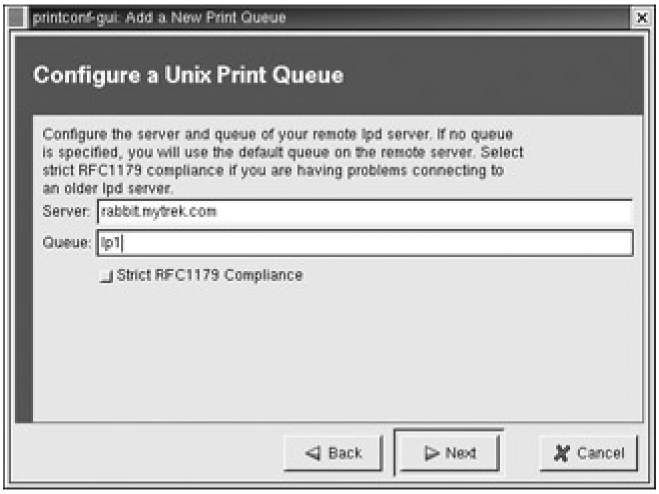
Rules will be saved using the service command with the **iptables** option, instead of **ipchains**. They will be saved to the **/etc/sysconfig/iptables** file.

```
service iptables save
```

Using Remote Printers

Once you have set up your network, you can use any Linux host to access printers connected to other hosts on the network, whether they be Linux, Windows, or Novell systems. The Linux host to which a printer is connected will first have to install that printer with printconf, as described in Chapter 4. Another Linux host can then access that printer by installing it as a remote printer. Once installed, the Linux host can print directly to that remote printer. For example, if an Epson printer is connected and installed on the **turtle.mytrek.com** host, the **rabbit.mytrek.com** host can install it as a remote printer, giving it a name of its own. Users on the **rabbit.mytrek.com** host can then print directly to the Epson printer connected on the **turtle** host. In fact, you could have all your printers connected to a single host and have all your other hosts print through it.

You use the printconf tool to install a remote printer. Select Printer Configuration from the Gnome system menu and click the New button. A series of dialogs will be displayed for entering the printer name, the type of remote printer, the printer's remote queue name, and the remote host the printer is connected to. For other Linux systems, entries are displayed for both the hostname and the IP address for the remote host (see Figure 7-9). Be sure to also enter the device driver and name information as described in Chapter 4.



printconf-gui: Add a New Print Queue

Configure a Unix Print Queue

Configure the server and queue of your remote lpd server. If no queue is specified, you will use the default queue on the remote server. Select strict RFC1179 compliance if you are having problems connecting to an older lpd server.

Server: rabbit.mytrek.com

Queue: lp1

☐ Strict RFC1179 Compliance

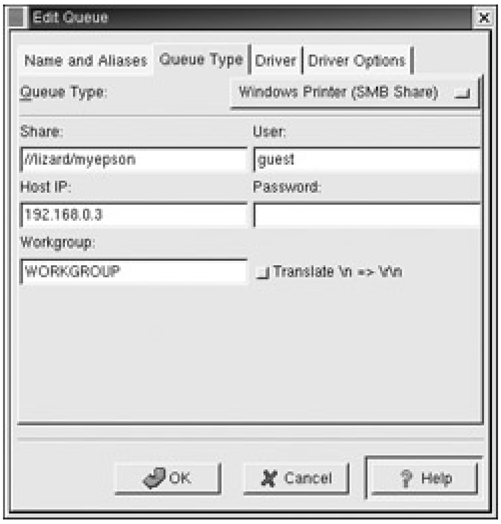
Figure 7-9: Remote Linux printer

The method works similarly with both Windows and Netware systems. However, to access Windows systems, you have to set up and run Samba. Samba will interface a Linux network with a Window network, allowing access to devices on Windows hosts like printers, CD-ROMs, and file systems. Samba is described in detail in Chapter 37.

To access a printer attached to a Windows system from a Linux host, you use the printconf tool on that Linux host to configure the printer as a remote Windows printer. In the printconf tool, select Queue and then select Windows Printer from the pop-up menu. Five entries are displayed. In the Share box, you enter the name of the Windows host, preceded by //, then followed by the name of the printer on that host, separated by a single slash. For example, the share name for a printer called **myepson** on the host **lizard** is

```
//lizard/myepson
```

Then enter the IP address of the Windows host. The workgroup is the workgroup that the Windows host belongs to. Though you can create passwords for each user, a simple approach is to just specify a guest user with no password. Be sure to set appropriate share and password permissions on the Windows host. Figure 7-10 shows remote Windows printer entries.



Edit Queue

Queue Type: Windows Printer (SMB Share)

Share: //lizard/myepson

User: guest

Host IP: 192.168.0.3

Password:

Workgroup: WORKGROUP

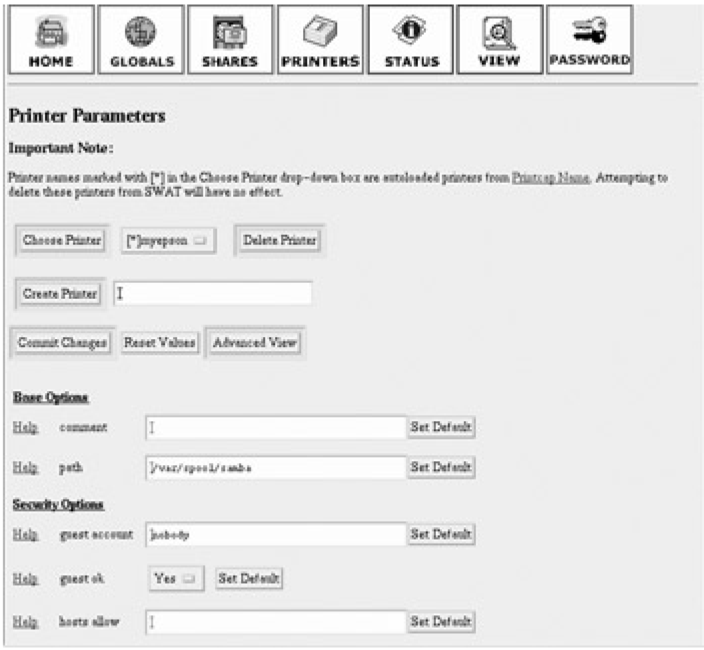
☐ Translate \n => \r\n

Figure 7-10: Remote Windows printer

If you want to allow Windows hosts to access a printer attached to a Linux system, you have to configure the printer in Samba. You need to first enable the use of swat with the following command:

```
chkconfig swat on
```

Then select Samba Configuration on the Gnome System menu. This opens a Web page on your browser with Samba buttons across the top. Click on the Printers button to display a page with buttons for selecting your printer. From the pop-up menu, select your printer, then click the Choose Printer button. Though you can set allowed users, a simple approach is to allow guest users to log in as the **nobody** user. To allow guest users, select Yes for "guest ok". Once you have made your changes, click on the Commit Changes button. Figure 7-11 shows a Samba printer page, where the printer **myepson** has been selected and guest access is allowed.



Printer Parameters

Important Note:

Printer names marked with [*] in the Choose Printer drop-down box are autoloaded printers from Printcap Name. Attempting to delete these printers from SWAT will have no effect.

[*]myepson

Base Options

Help comment

Help path /var/spool/samba

Security Options

Help guest account nobody

Help guest ok Yes

Help hosts allow

Figure 7-11: Samba printers

Once you have made changes, you can have them take effect by restarting Samba with the following command:

```
service smb restart
```

Note Macintosh printers on Macintosh systems are connected using a different network protocol called NetaTalk (see Chapter 32).

Setting Up a Web Server

For a standard installation, Red Hat installs the Apache Web server. Any user on your network can access the Web server using the name of the host it is running on. For this example, the Web server is running on the gateway host, **turtle.mytrek.com**. Normally, an alias is set up that is used for the hostname, such as **www.mytrek.com**. Entering this address in a Web browser will display the main document page of the Apache Web server. Web pages for your site are kept in **/var/www/html**. You can add your own pages here, building your Web site. The Apache manual will already be there.

Part III: Red Hat Desktop Workstation

Chapter List

Chapter 8: Gnome

Overview

The GNU Network Object Model Environment, also known as *Gnome,* is a powerful and easy-to-use environment consisting primarily of a panel, a desktop, and a set of GUI tools with which program interfaces can be constructed. Gnome is designed to provide a flexible platform for the development of powerful applications. Currently, Gnome is strongly supported by Red Hat and is its primary GUI interface, though Gnome is also provided by most other distributions. Gnome is completely free under the GNU Public License. You can obtain the source directly, as well as documentation and other Gnome software, from the Gnome Web site at **www.gnome.org**. Several companies have joined together to form the Gnome Foundation, an organization dedicated to coordinating the development of Gnome and Gnome software applications. These include such companies as Sun, IBM, and Hewlett-Packard as well as Linux distributors like Red Hat, Caldera, and TurboLinux along with Gnome developers such as Ximian. Modeled on the Apache Foundation, which developed the Apache Web server, the Gnome Foundation will provide direction to Gnome development as well as organizational financial, and legal support. Recently Sun announced that it was adopting Gnome as the desktop interface for its Solaris operating system, replacing the Motif-based Common Desktop Environment (CDE).

The core components of the Gnome desktop consist of a panel for starting programs and desktop functionality. Other components normally found in a desktop, such as a file manager, Web browser, and window manager, are provided by Gnome-compliant applications. Gnome provides libraries of Gnome GUI tools that developers can use to create Gnome applications. Programs that use buttons, menus, and windows that adhere to a Gnome standard can be said to be Gnome-compliant. For a file manager, the Gnome desktop currently uses a Gnome version of Midnight Commander-and will soon replace it with Nautilus. The Gnome desktop does not have its own window manager as KDE does. The Gnome desktop uses any Gnome-compliant window manager. Currently, the Sawfish window manager is the one bundled with the Gnome distributions.

Integrated into Gnome is support for component model interfaces, allowing software components to interconnect regardless of the computer language in which they are implemented or the kind of machine on which they are running. The standard used in Gnome for such interfaces is the Common Object Request Broker Architecture (CORBA), developed by the Object Model Group for use on Unix systems. Gnome uses the ORBit implementation of CORBA. With such a framework, Gnome applications and clients can directly communicate with each other, enabling you to use components of one application in another. With Gnome 2.0, Gnome will officially adopt GConf and its libraries as the underlying method for configuring Gnome and its applications. GConf can configure independently coordinating programs such as those that make up the Nautilus file manager.

You can find out more about Gnome at its Web site at **www.gnome.org**. This site not only provides a detailed software map of current Gnome projects with links to their development sites, it also maintains extensive mailing lists for Gnome projects to which you can subscribe. The Web site provides online documentation, such as the Gnome User's Guide and FAQs. If you want to develop Gnome programs, check the Gnome developer's Web site at **developer.gnome.org.** The site provides tutorials, programming guides, and development tools. Here you can find the complete API reference manual online, as well as extensive support tools such as tutorials and Interactive Development Environments (IDE). The site also includes detailed online documentation for the GTK+ library, Gnome widgets, and the Gnome desktop.

A new file manager for the Gnome desktop, called Nautilus, has been released (originally developed by Eazel). With Gnome 1.4, Nautilus officially replaces the GNU Midnight Commander file manager now used on Gnome desktop. Nautilus is designed to operate as a desktop shell that can support numerous components, letting you operate a Web browser within it or decompress files. Nautilus development is now being carried on independently. You can download Nautilus source from the Gnome CVS depository or from **www.ximian.com**. You can find out more about nautilus from the Nautilus User's Manual that is part of the Gnome 1.4 User's Guide at **www.gnome.org**.

Note An enhanced version of Gnome known as Ximian Gnome can be downloaded from Ximian at **www.ximian.com**. There are versions for most distributions, including Red Hat.

GTK+

GTK+ is the widget set used for Gnome applications. Its look and feel was originally derived from Motif. The widget set is designed from the ground up for power and flexibility. For example, buttons can have labels, images, or any combination thereof. Objects can be dynamically queried and modified at runtime. It also includes a theme engine that enables users to change the look and feel of applications using these widgets. At the same time, the GTK+ widget set remains small and efficient.

The GTK+ widget set is entirely free under the Library General Public License (LGPL). The LGPL enables developers to use the widget set with proprietary, as well as free, software (GPL would restrict it to just free software). The widget set also features an extensive set of programming language bindings, including C++, Perl, Python, Pascal, Objective C, Guile, and Ada. Internalization is fully supported, permitting GTK+-based applications to be used with other character sets, such as those in Asian languages. The drag-and-drop functionality supports both Xdnd and Motif protocols, allowing drag-and-drop operations with other widget sets that support these protocols, such as Qt and Motif.

The Gnome Interface

The Gnome interface consists of the panel and a desktop, as shown in Figure 8-1. The panel appears as a long bar across the bottom of the screen. It holds menus, programs, and applets. An *applet* is a small program designed to be run within the panel. On the panel is a button with a large bare foot imprint on it. This is the Gnome applications menu, the main menu. The menu operates like the Start menu in Windows, listing entries for applications you can run on your desktop. You can display panels horizontally or vertically, and have them automatically hide to show you a full screen.

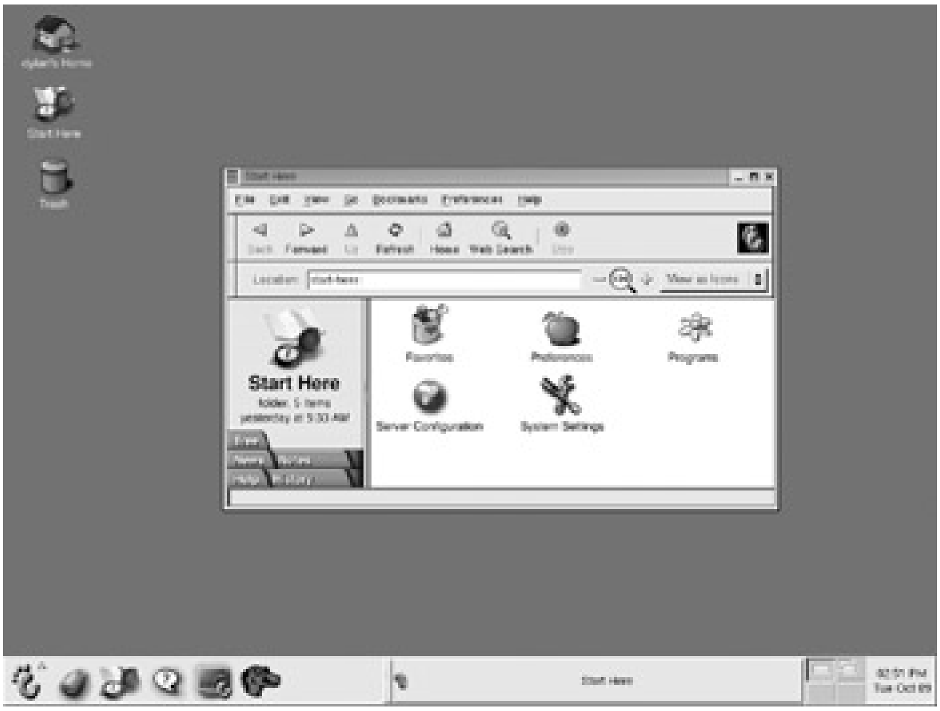


Figure 8-1: Gnome

The remainder of the screen is the desktop. Here, you can place directories, files, or programs. You can create them on the desktop directly or drag them from a file manager window. A click-and-drag operation with the middle mouse button enables you to create links on the desktop to installed programs. Initially, the desktop only holds an icon for your home directory. Clicking it opens a file manager window to that directory. A right- click anywhere on the desktop displays a desktop menu (see Table 8-1) with which you can open new windows, create new folders, and mount floppy disks and CD-ROMs.

Table 8-1: The Gnome 1.4 Desktop Menu (Nautilus)

Menu Item	**Description**
New Window	Starts a new Nautilus file manager window on your desktop, showing your home directory.
New Terminal	Launches a new Gnome terminal window that navigates to the desktop directory.
New Folder	Creates a new directory on your desktop.
Clean Up by Name	Arranges your desktop icons.
Disks	Displays submenu that lists floppy and CD-ROM devices that you

Table 8-1: The Gnome 1.4 Desktop Menu (Nautilus)

Menu Item	Description
	can select to mount. Mounted disks will appear as CD-ROM or floppy icons on your desktop, which you can use to access them or unmount later.
Change Desktop Background	Opens Gnome Control Center with the Background caplet selected to let you select a new background for your desktop.

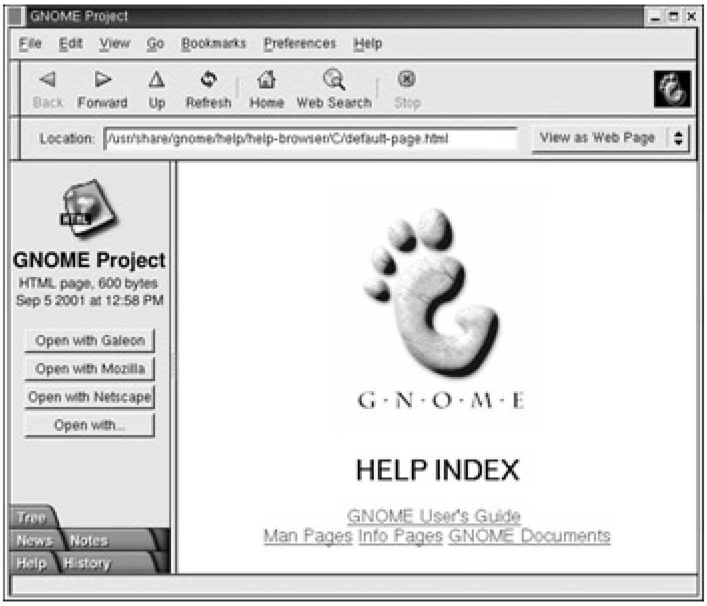
From a user's point of view, you can think of the Gnome interface as having four components: the desktop, the panel, the main menu, and the file manager. In its standard default configuration, the Gnome desktop displays a Folder icon for your home directory in the upper-left corner. Some distributions may include other icons, such as links to the Gnome Web site or to the Linux Documentation site. Initially, a file manager window opens on the desktop, displaying your home directory. The panel has several default icons: The main menu (Bare Foot), a screen lock feature (a padlock), the terminal program (Monitor), the Gnome Help System (Question Mark), the Start Here window (Pocket compass), the Gnome pager (Squares), and a clock. Red Hat 7.2 includes Mozilla.

To start a program, you can select its entry in the main menu, click its application launcher button in the panel (if there is one), double-click its icon in either the desktop or the file manager window, drag a data file to its icon, or select the Run Program entry in the main menu. This opens a small window where you can type in the program name.

When you first start Gnome, the Start Here window is displayed. From here, you can access your favorite Web sites and files (Favorites), select and run applications (Programs), customize your Gnome desktop (Preferences), and perform administrative tasks for both your system and your servers (Server Configuration and System Settings). Double-clicking an icon opens a window listing icons for sub-windows or tools. In effect, the Start Here window is mimicking the Main menu; you can select and run applications from the Start Here's Program window, just as you can from the Main menu's Program menu. The Preferences window lists Gnome configuration tools (capplets) for setting up your Gnome preferences. In effect, it replaces the Gnome Control Center window used in previous versions. See the Gnome Configuration section for more details. If you need to configure administrative tasks, such as setting up network connections or managing servers, you can choose the Server Configuration or System Settings windows. Most of those tools are accessible only by the root user.

To quit Gnome, you select the Logout entry in the main menu or click the terminal icon displaying the moon (on Ximian Gnome you choose the Logout entry in the Menu panel's System menu). You can also add a Logout button to the panel, which you could use instead. To add the Logout button, right-click the panel and select the Add Logout Button entry. A Logout button then appears in the panel. When you log out, the Logout dialog box is displayed. You have three options. The first option, Logout, quits Gnome, returning you to the login window (or command line shell still logged into your Linux account, if you started Gnome with **startx**). The second option, Halt, not only quits Gnome, but also shuts down your entire system. The third option, the Reboot entry, shuts down and reboots your system. The Logout entry is selected by default. Halt and Reboot are only available to the root user. If normal users execute them, they are prompted to enter the root user password to shut down. You can also elect to retain your desktop by clicking the "Save current setup" check box. This reopens any programs or directories still open when you logged out. Gnome-compliant window managers also quit when you log out of Gnome. You then must separately quit a window manager that is not Gnome-compliant after logging out of Gnome.

The Gnome Help system, shown in Figure 8-2, provides a browser-like interface for displaying the Gnome user's manual, Man pages, and info documents. It features a toolbar that enables you to move through the list of previously viewed documents. You can even bookmark specific items. A Web page interface enables you to use links to connect to different documents. You can easily move the manual or the list of Man pages and info documents. You can place entries in the location box to access specific documents directly. Special URL-like protocols are supported for the different types of documents: ghelp: for Gnome help, man: for man pages, and info: for the info documents.



Location: /usr/share/gnome/help/help-browser/C/default-page.html

Figure 8-2: The Gnome Help system

The Gnome Desktop

The Gnome desktop provides you with all the capabilities of GUI-based operating systems (see Figure 8-3). You can drag files, applications, and directories to the desktop, and then back to Gnome-compliant applications. If the desktop stops functioning, you can restart it by starting the Gnome file manager (Nautilus). The desktop is actually a backend process in the Gnome file manager. But you needn't have the file manager open to use the desktop. The Gnome 1.4 desktop with Nautilus is described in this section.

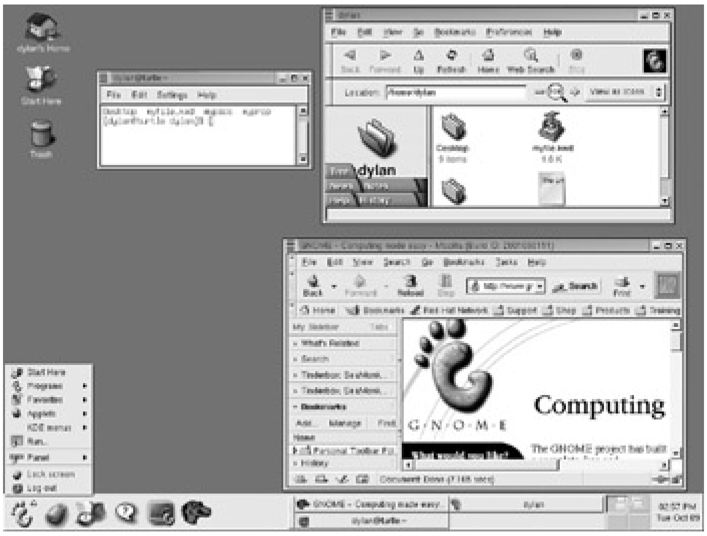


Figure 8-3: The Gnome 1.4 desktop

Although the Gnome desktop supports drag-and-drop operations, these work only for applications that are Gnome-compliant. You can drag any items from a Gnome-compliant application to your desktop, and vice versa. Any icon for an item that you drag from a file manager window to the desktop also appears on the desktop. However, the default drag-and-drop operation is a **move** operation. If you select a file in your file manager window and drag it to the desktop, you are actually moving the file from its current directory to the Gnome desktop directory, which is located in your home directory and holds all items on the desktop (notice this is a dot file). For Gnome 1.2, the desktop directory is **.gnome-desktop**. In the case of dragging directory folders to the desktop, the entire directory and its subdirectories would be copied to the Gnome desktop directory.

In most cases, you only want to create on the desktop another way to access a file without moving it from its original directory. You can do this by creating a link or a program launcher, instead of moving the file. To create a link, click and drag the file while holding down the SHIFT key. A copy of the icon then appears with a small arrow in the right corner indicating it is a link. You can then click this link to start the program, open the file, or open the directory, depending on what kind of file you linked to.

You can then use that icon to access the item directly. This is often used for starting common programs. For example, you can SHIFT-click and drag the Netscape icon to the desktop to create a Link icon for Netscape. Double-clicking the icon starts Netscape. You can do the same with files. In this case, their respective program is started. If the item is a directory, the file manager starts up opened to that directory. If you want to have an application placed on your desktop that is not Gnome-compliant, you can manually place a link in your home directory's Gnome desktop directory.

You can also copy a file to your desktop by clicking-and-dragging it from a file manager window to your desktop, and then pressing the CTRL key before you release the left mouse button. You will see the small arrow in the upper right-hand corner of the copied icon change to a + symbol, indicating that your are creating a copy instead of a link.

As an alternative to the desktop, you can drag any program, file, or directory to the panel; a launcher applet is then automatically created for it on the panel. The item is not moved or copied. You can also right-click anywhere on the empty desktop to display a menu. You will notice entries for a New Folder. Remember, this entry creates a new directory on your desktop, specifically in the Gnome desktop directory. The entries for this menu are listed in Table 8-1.

The desktop also displays icons for any drives you have mounted, such as a CD-ROM or floppy drives, provided they are user mountable. Nautilus automatically mounts CD-ROMs when you insert them into your CD-ROM drive, displaying the CD-ROM icon and opening a Nautilus window displaying the CD-ROM's contents.

You can manually mount a CD-ROM or floppy disk by right-clicking anywhere on the desktop to display the desktop menu and then selecting the CD-ROM or Floppy entry in the Disks menu. An icon for that device will appear with the name of the CD-ROM or floppy disk. You can then access the disk in the CD-ROM drive either by double-clicking it or right-clicking and selecting the Open entry. A file manager window opens to display the contents of the CD-ROM disk. To unmount a CD-ROM, right-click the CD-ROM icon and select the Unmount entry. The CD-ROM disk is automatically ejected. The same procedure works for floppy disks, using the Floppy Disk icon. Be sure you don't remove a mounted floppy disk until you have first unmounted it, selecting the Unmount entry in the pop-up menu.

Usually a window manager extends a desktop into several areas that appear as different screens. Gnome's drag-and-drop operation works on desktop areas provided by a Gnome-compliant window manager. Gnome does not directly manage desktop areas, though you can use the Gnome pager to move to them. You use the window manager configuration tool to configure them-in this case Sawfish, which is the default window manager for Gnome used in Red Hat. In addition, most window managers, including Sawfish, also support virtual desktops. Instead of being extensions of the same desktop area, virtual desktops are separate entities. The Gnome pager on the Panel supports virtual desktops, creating icons for each in the panel, along with task buttons for any applications open on them. You can use the Gnome pager to move to different virtual desktops and their areas.

Window Managers

Gnome works with any window manager. However, desktop functionality, such as drag-and-drop capabilities and the Gnome pager, only work with window managers that are Gnome-compliant. The current release of Gnome uses the Sawfish window manager. Its is completely Gnome-compliant and is designed to integrate with the Gnome desktop without any duplication of functionality. However, other window managers such as Enlightenment, FVWM, IceWin, and Window Maker can also be used. Check a window manager's documentation to see it is Gnome-compliant.

Sawfish employs much the same window operations as used on other window managers. You can resize a window by clicking any of its sides or corners and dragging. You can move the window with a click-and-drag operation on its title bar. You can also right-click and drag any border to move the window, as well as ALT-click anywhere on the window. The upper-right corner lists the Maximize, Minimize, and Close buttons. If the Gnome pager is running in your panel, then Minimize creates a button for the window in the panel that you can click to restore it. If the Gnome pager is not present, the window will iconify, minimizing to an icon on the desktop. You can right-click on the title bar of a window to display a window menu with entries for window operations. These include a desktop entry to move the window to another desktop area and the Stick option, which displays the window no matter to what desktop area you move.

You can also access the Sawfish desktop menu. To display the menu, middle-click anywhere on the desktop (hold both mouse buttons down at the same time for a two- button mouse). A pop-up menu then appears with submenus for Gnome, user, and other applications, as well as the Desktop, Themes, and Sawfish configurations. You can use this menu to start any application, if you want. With the desktop menus, you move to different desktop areas and virtual desktops. The Themes menu enables you to choose different Sawfish themes (these are separate from KDE themes). Sawfish also has extensive configuration options, discussed in a later section.

If you have several window managers installed on your system, you can change from one to the other using the Window Manager capplet. *Capplet* is the term used for a *control applet,* a module used to configure your desktop. Select the entry in the main menu to start the Window Manager Settings menu, or select its icon in the Desktop window, which you open from Preferences in the Start Here window. A panel is displayed listing your window managers. To add others to the list, click the Add button on the right side of the panel. This opens a window that prompts you to enter an identifying name for the window manager, the command that starts the window manager, and any configuration tool it may use. If the window manager is Gnome-compliant, you can click the button Window Manager Is Session Managed. Once you finish making your entries and click OK, the new window manager appears in the list on the Window Manager panel. Select it and click Try to run that window manager. If you want to run the window manager's configuration tool, click the Run Configuration Tool button.

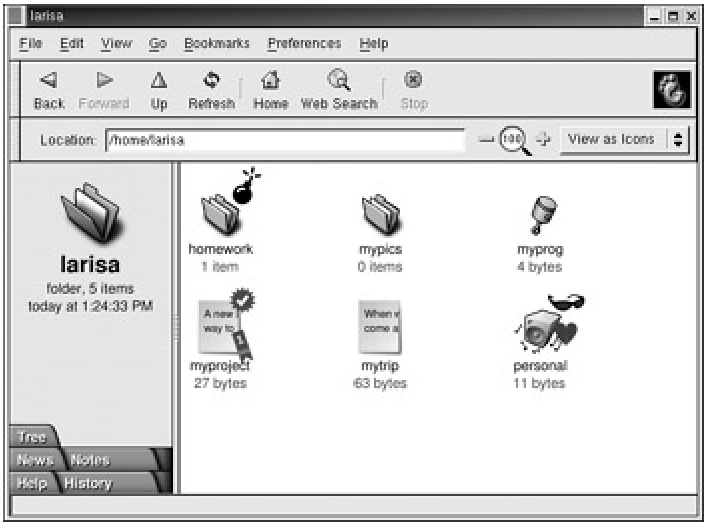
The Gnome (1.4) File Manager: Nautilus

With Gnome 1.4, the file manager for Gnome was changed from Gnome Midnight Commander (GMC) to Nautilus (Red Hat 7.1 still uses GMC). Nautilus supports the standard features for copying, removing, and deleting items as well as setting permissions and displaying items just as GMC does. Nautilus also provides enhancements such as zooming capabilities, user levels, and theme support. You can enlarge or reduce the size of your file icons, select from novice, intermediate, or expert levels of use, and customize the look and feel of Nautilus with different themes. Nautilus also lets you set up customized views of file listings, enabling you to display images for directory icons and run component applications within the file manager window. For example, a directory of MP3 files could have an album cover for its directory icon and run a component MP3 player within the file manager window to play a file.

Nautilus was designed as a desktop shell in which different components can be employed to add functionality. For example, within Nautilus, a Web browser can be executed to provide Web browser capabilities within a Nautilus file manager window. Nautilus is not only a file manager, but also a desktop shell based on a component architecture. Different components, such as a Web browser, compression commands, or an image viewer, can be used to add capabilities to this desktop shell.

Note Earlier versions of Linux, including Red Hat 7.1, still use Gnome 1.2 with the Gnome Midnight Commander (GMC) file manager, but are compatible with Gnome 1.4 and Nautilus.

The standard Nautilus window displays a menu bar and toolbar of file manager commands along with a Location box at the top (see Figure 8-4). The rest of the window is divided into two panes. The left pane is a sidebar used to display information about the current working directory. The right pane is the main panel that displays the list of files and subdirectories in the current working directory. A status bar at the bottom of the window displays information about a selected file or directory.


Location: /home/larisa

View as Icons

Figure 8-4: Nautilus file manager

With the preferences menu, you can set your level of expertise: advanced, intermediate, or beginner. The different levels allow for simpler methods of managing files. If you only need the basic file management capabilities, you can choose beginner, leaving advanced and intermediate for the more complex tasks. Next to the Location box is an element for zooming in and out the view of the files. Click the + button to zoom in and the - button to zoom out. Next to the zoom element is a drop-down menu for selecting the different views for your files such as icons, small icons, or details.

The sidebar has five different tabbed views for displaying additional information about files and directories: Tree, History, News, Help, and Notes. The Tree view will display a tree-based hierarchical view of the directories and files on your system, highlighting the one you have currently selected. You can use this tree to move to other directories and files. The tree maps all the directories on your system, starting from the root directory. You can expand or shrink any directory by clicking the + or - symbol before its name. Select a directory by clicking the directory name. The contents of that directory are then displayed in the main panel. The History tab shows previous files or directories you have accessed, handy for moving back an forth between directories or files. The Help tab provides access to documentation such as Gnome manuals, info pages, and Man documentation. On the Notes tab, you can enter notes about your current working directory. The News tab will display breaking news from sites you have selected. Click the Select Site button to display a list of news sites such as CNN, Gnome News, or Linux Today. URLs for different stories will appear in the sidebar when the News panel is selected.

You can view a directory's contents as icons, as a detailed list, as music, or as a custom (other) view. You select the different options from the pop-up menu located on the right side of the Location bar. The List view provides the name, permissions, size, date, owner, and group. For a custom view, you can select the informational fields you want displayed for your files. In the List views, buttons are displayed for each field across the top of the main panel. You can use these buttons to sort the lists according to that field. For example, to sort the files by date, click the Date button; to sort by size, click the Size button.

In the Icon view you can sort icons, change their sizes, and even preview their contents without opening them. To sort items in the Icon view, select the Lay Out Items entry in the View menu, and then select a layout option. To change an icon's size, select Stretch Icon. Handles will appear on the icon image. Click and drag the handles to change its size. To restore the icon, select Restore Icon's Original Size in the Edit menu. Certain types of files have their icons display previews of their contents. Image files will have their icon display a small version of the image. A music file like an MP3 file will start playing when the mouse pointer moves over its icon. A text file will display in its icon the first few words of its text.

The music view lets you treat directories of MP3 files as if they were your own digital music albums. When you select the music view, only MP3 files are displayed. Clicking on a file starts the music player, which will automatically play from track to track. You can even select a cover image for your music directory icon.

You can click anywhere on the main panel to display a pop-up menu with entries for managing and arranging your file manager icon (see Table 8-2). To create a new folder, select New Folder, and to open a new file manager window, select New Window. The Lay Out Items entry displays a submenu with entries for sorting your icons by name, size, type, date, or even emblem. The Manually entry lets you move icons wherever you want on the main panel. The Clean up by Name entry will display your icons alphabetically. The Zoom in entry will enlarge your view of the window, making icons bigger, and Zoom out will reduce your view, making them smaller. Normal Size restores them to the standard size. Reset Background lets you change the background used on the File Manager window, useful for music folders where you display your favorite album cover. You can also cut, copy, or paste files to let you more easily move or copy them between folders.

Table 8-2: Nautilus File Manager Menu

Menu Item	**Description**
New Window	Open a file or directory in a separate window.
New Folder	Create a new subdirectory in the directory.
Lay Out Items	Displays a submenu to arrange files by name, size, type, date, or emblems.
Clean Up by Name	Orders files alphabetically.
Cut, Copy, Paste	Cuts, copies, and pastes files to let you more easily move or copy them between folders.
Zoom in	Close-up view of icons, making them appear larger.
Zoom out	Distant view of icons, making them appear smaller.
Normal Size	Restores view of icons to standard size.
Reset Background	Sets the background image for the file manager main panel.

Note The Lay Out Item submenu is not provided in the Nautilus pop-up menu when the main panel displays files in the list mode.

The Nautilus file manager operates similarly to a Web browser. It maintains a list of previously viewed directories; you can move back and forth through that list using the toolbar buttons. The LEFT ARROW button moves you to the previously displayed directory, and the RIGHT ARROW button moves you to the next displayed directory. The UP ARROW button moves you to the parent directory, and the Home button moves you to your home directory. To use a pathname to go directly to a given directory, you can type the pathname in the Location box and press ENTER.

To search for files or directories, click the Find icon on the toolbar if there is one (see following Note). Depending on the expertise level you set, Nautilus can perform simple or complex searches (simple for beginners and complex for advanced and intermediate). The Location box is replaced by the Find box, where you can enter the pattern you want to search for. Then click on the Find Them button to the right. Complex searches let you specify different attributes such as type, owner, or even file contents. For complex searches, two pop-up menus are added with which you can further refine your search. These menus specify criteria such as searches by name, content, file type, size, date modified, and owner. Nautilus searches make use of a search daemon called **Medusa** that creates an index of all your files, enabling very fast searches. **Medusa** even indexes the contents of your file, supporting text searches of file contents. (For **Medusa** to work, the **crond** daemon must be running.)

Note Red Hat 7.2 with version 1.0 for Nautilus does not include support for Find operations. A Find icon is not displayed. If you are using Ximian Gnome, you can use a Find utility located on the System menu in the Menu panel. It has many of the same features as the Nautilus Find operation.

To open a subdirectory, you can double-click its icon or single-click the icon and select Open from the File menu. If you want to open a separate Nautilus window for that directory, right-click the directory's icon and select Open in a New Window.

As a Gnome-compliant file manager, Nautilus supports GUI drag-and-drop operations for copying and moving files. To move a file or directory, click and drag from one directory to another as you would on Windows or Mac interfaces. The **move** operation is the default drag-and-drop operation in Gnome. To copy a file, click and drag with the right mouse button (not the left) and select Copy Here from the pop-up menu. You can also click and drag normally and then press the CTRL key before you lift up on the left mouse button.

Note If you move a file to a directory on another partition (file system), it will be copied instead of moved.

You can also perform remove, duplicate, and link creation operations on a file by right-clicking its icon and selecting the action you want from the pop-up menu that appears (see Table 8-3). For example, to remove an item, right-click it and select the Move to Trash entry from the pop-up menu. This places it in the Trash directory where you can later delete it by selecting Empty Trash from the Nautilus File menu. To create a copy of a file, you can select Duplicate from the pop-up menu to create a duplicate version in the same directory. The name of the copy will begin with the term "Duplicate". To create a link, right-click the file and select Create Link from the pop-up menu. This creates a new link file that begins with the term "Link" (see Figure 8-5 for an example of the Duplicate, Link, and Rename operations).

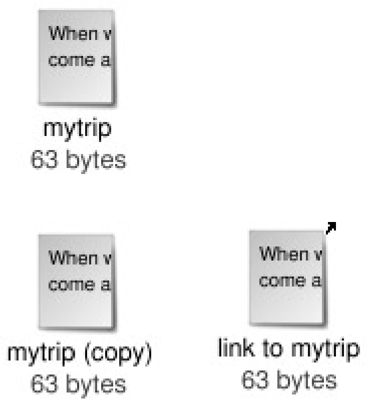


Figure 8-5: Nautilus duplicate, link, and rename operations

Table 8-3: The Nautilus File Pop-Up Menu

Menu Item	Description
Open	Open the file with its associated application. Directories are opened in the file manager.
Open in a New Window	Open a file or directory in a separate window.
Open With	Select an application with which to open this file. A submenu of possible applications is displayed.
Show Properties	Display the Properties dialog box for this file. There are three panels: Statistics, Options, and Permissions.
Move to Trash	Move a file to the Trash directory, where you can later delete it.
Duplicate	Create a duplicate copy of the file in the same directory.
Make Link	Create a link to that file in the same directory.
Rename	Rename the file.
Stretch Icon	Change the size of a selected icon.
Remove Custom Image	Remove a custom image you selected for the icon.
Restore Icon's Original Size	A selected icon you enlarged earlier is restored to its standard size.

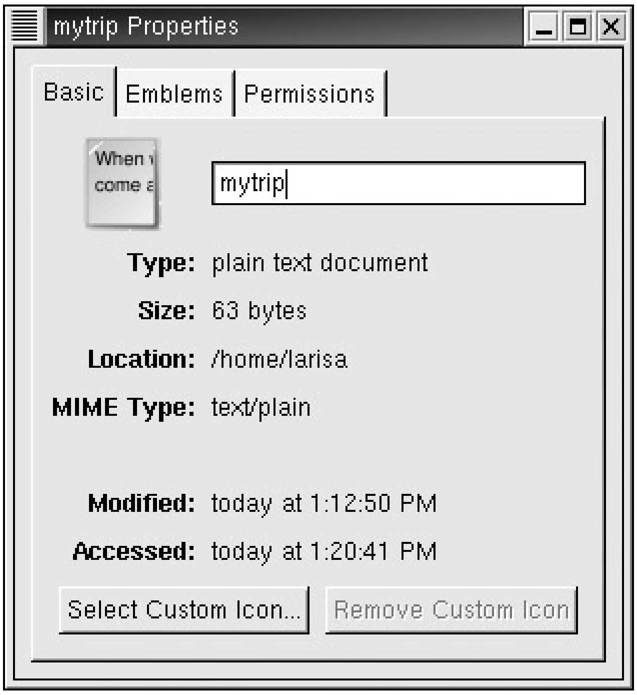
To rename a file you can either right-click on the file's icon and select the Rename entry from the pop-up menu or slowly click on the name of the file shown below its icon. The name of the icon will be highlighted in a black background, encased in a small text box. You can then click on the name and delete the old name, typing a new one. You can also rename a file by entering a new name in its Properties dialog box. Use a right-click and select Show Properties from the pop-up menu to display the Properties dialog box. On the General tab, you can change the name of the file.

File operations can be performed on a selected group of files and directories. You can select a group of items in several ways. You can click the first item and then hold down the SHIFT key while clicking the last item. You can also click and drag the mouse across items you want to select. To select separated items, hold the CTRL key down as you click the individual icons. If you want to select all the items in the directory, choose the Select All entry in the Edit menu. You can also select files based on pattern matches on the filenames. Choose the

Select Files entry in the Edit menu. You can then enter a pattern using Linux file-matching wildcard symbols such as * (See Chapter 11). For example, the pattern ***.c** would select all C source code files (those ending with the extension ".c"). You can then click and drag a set of items at once. This enables you to copy, move, or even delete several files at once. To move files between directories, open two file manager windows to the respective directories. Then click and drag the items from one window to the other.

You can start any application in the file manager by double-clicking either the application itself or a data file used for that application. If a file does not have an associated application, you can right-click the file and select the Open With entry. A submenu displays a list of possible applications. If your application is not listed, you can select Other application to open a dialog box where you can choose the application with which you want to open this file. You can also use a text viewer to display the bare contents of a file within the file manager window. Drag-and-drop operations are also supported for applications. You can drag a data file to its associated application icon (say, one on the desktop); the application then starts up using that data file.

With the Properties dialog box, you can view detailed information on a file and set options and permissions (see Figure 8-6). A Properties box has three panels: Basic, Emblems, and Permissions. The *Basic panel* shows detailed information such as type, size, location, and date modified. The type is a MIME type, indicating the type of application associated with it. The file's icon is displayed at the top with a text box showing the file's name. You can edit the filename in this text box, changing that name. A button at the bottom labeled Select Custom Icon will open a dialog box showing available icons you can use. You can select the one you want from that window. The Remove Custom Icon button will restore the default icon image. The *Emblems panel* enables you set the emblem you want displayed for this file, displaying all the emblems available. The *Permissions panel* shows the read, write, and execute permissions for user, group, and other, as set for this file. You can change any of the permissions here, provided the file belongs to you. The panel will also show the file's owner and its group. The group name expands to a pop-up menu listing different groups, allowing you to select one to change the file's group.



mytrip

Type: plain text document

Size: 63 bytes

Location: /home/larisa

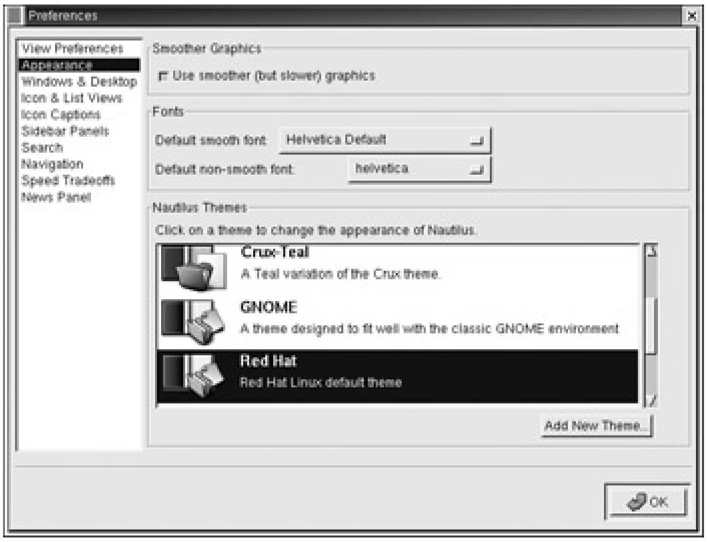
MIME Type: text/plain

Modified: today at 1:12:50 PM

Accessed: today at 1:20:41 PM

Figure 8-6: File properties on Nautilus

You can set preferences for your Nautilus file manager in the Preferences dialog box. Access this dialog box by selecting the Edit Preferences item in the Preferences menu. The Preferences dialog box shows a main panel with a sidebar with several configuration entries, including Appearance, Windows & Desktop, Icon & List Views, Sidebar panels, Search, Navigation, and Speed tradeoffs (see Figure 8-7). You use these dialog boxes to set the default display properties for your Nautilus file manager. For example, Windows & Desktop allows you to select which bars to display by default, such as the sidebar or the toolbar. Appearance lets you select the style you want to use. For the Sidebar panel, you can select which tab to display, such as history or tree. On the Search menu, you can specify the default Web search site to use.



□ Use smoother (but slower) graphics

Default smooth font: Helvetica Default

Default non-smooth font: helvetica

Click on a theme to change the appearance of Nautilus.

Crux-Teal
A Teal variation of the Crux theme.

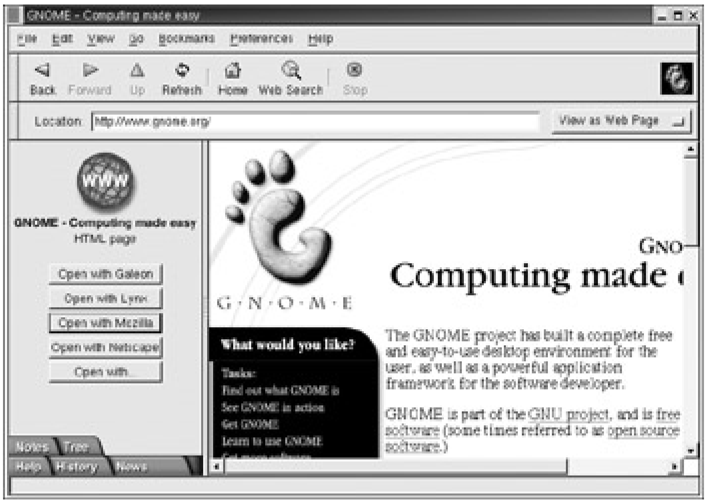
GNOME
A theme designed to fit well with the classic GNOME environment

Red Hat
Red Hat Linux default theme

Figure 8-7: Nautilus Preferences dialog box

Nautilus as a Web Browser

Nautilus is also an operational Web browser (see Figure 8-8). You can use the Location box to access any Web or FTP site. Just enter the URL for the Web site in the Location box and press ENTER (you do not need to specify www or http://). However, Nautilus is not a fully functional Web browser. When you access a page, it will display buttons in the sidebar to open the page using one of several Web browsers installed on your system, such as Netscape, Mozilla, or Lynx. The Go menu and History tab in the sidebar maintain a history list of Web sites you have previously accessed. You can also right-click on the Back and Forward buttons to display this history list. To clear the history list, select Forget History in the Go menu. Nautilus also supports Bookmarks, which can be displayed and edited using the Bookmarks menu. Clicking on the Web Search icon will open the page of your favorite Web search engine, such as Yahoo or Google.



The GNOME project has built a complete free and easy-to-use desktop environment for the user, as well as a powerful application framework for the software developer.

GNOME is part of the GNU project, and is free software (some times referred to as open source software.)

Figure 8-8: Nautilus as a Web browser

Note Nautilus is also Internet-aware. You can use the Location box to access an FTP site and display the directories on that remote site, and then drag and drop files to another file manager window to download them to your system. Be sure to include the FTP protocol specification, **ftp://**.

The Gnome Panel

The *panel* is the center of the Gnome interface (see Figure 8-9). Through it you can start your applications, run applets, and access desktop areas. You can think of the Gnome panel as a type of tool you can use on your desktop. You can have several Gnome panels displayed on your desktop, each with applets and menus you have placed in them. In this respect, Gnome is flexible, enabling you to configure your panels any way you want. You can customize a panel to fit your own needs, holding applets and menus of your own selection. You may add new panels, add applications to the panel, and add various applets.

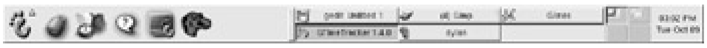


Figure 8-9: The Gnome panel

You can hide the panel at any time by clicking either of the Hide buttons located on each end of the panel. The Hide buttons are thin buttons showing a small arrow. This is the direction in which the panel will hide. To redisplay the panel, move your mouse off the screen in that direction at the bottom of the screen. If you want the panel to automatically hide when you are not using it, select the Auto-Hide option in the panel configuration window. Moving the mouse to the bottom of the screen redisplays the panel. You can also move the panel to another edge of the screen by clicking and dragging on either end of the panel with your middle mouse button (both buttons simultaneously for two-button mice).

To add a new panel, select the Create Panel entry in the Panel menu. You can then select the panel type, choosing from a menu, edge, floating, sliding, or aligned panel. The default is an edge panel. An *edge panel* is displayed across one of the edges of the screen. Your original panel is an edge panel. A *menu panel* is a panel implemented as a menu bar across the top of the desktop with menus for Programs, Favorites, Settings, and Desktop. It can hold any objects that a standard panel can hold. A floating panel is one that you can position anywhere on the desktop. A *sliding panel* is sized to the number of items in the panel and can be placed anywhere on the edge of the desktop. An *aligned* panel is a smaller panel also sized to the number of items on it and is positioned in the center of the edge of a desktop. You can change a panel's type at any time by right-clicking the panel and selecting the alternate configuration. Figure 8-10 shows examples for the different types of Gnome panels.

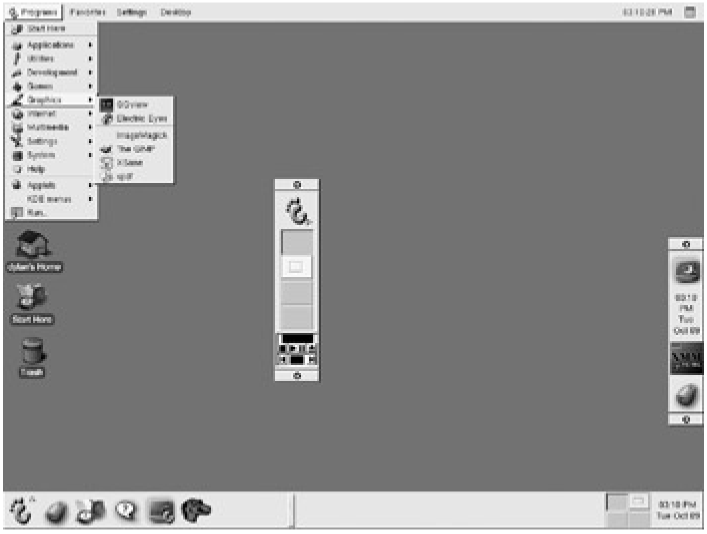


Figure 8-10: Gnome panel types

A panel can contain several different types of objects. These include menus, launchers, applets, drawers, and special objects. The main menu is an example of a panel menu. You can create your own, or even drag submenus down to the panel from the main menu. Launchers are buttons used to start an application or execute a command. The Netscape icon is an example of a launcher button. You can drag any application entry in the main menu to the panel and create a launcher for it on the panel. An applet is a small application designed to run within the panel. The Gnome Desk Guide showing the different desktops is an example of a Gnome applet. A drawer is an extension of the panel that can be open or closed. You can think of a drawer as a shrinkable part of the panel. You can add anything to it that you can to a regular panel, including applets, menus, and even other drawers. Special objects are used for special tasks not supported by other panel objects. For example, the Logout and Lock buttons are special objects.

Adding Applications and Applets

Adding applications to a panel is easy. For an application already in the main menu, you only need to go to its entry and right-click it. Then select the Add This Launcher To Panel entry.

An application launcher for that application is then automatically added to the panel. Suppose you use gEdit frequently and want to add its icon to the panel, instead of having to go through the main menu all the time. Right-click the gEdit menu entry and select the Add This Launcher To Panel option. The gEdit icon now appears in your panel.

To add an application icon not in the Main menu, first right-click the panel to display the pop-up menu and select the Add New Launcher entry. This opens the Create Launcher Applet window for entering properties for the applications launcher. You are prompted for the application name, the command that invokes it, and its type. To select an icon for your launcher, click the Icon button. This opens the Icon Picker window, listing icons from which you can choose.

You can also group applications under a Drawer icon. Clicking the Drawer icon displays a list of the different application icons you can then select. To add a drawer to your panel, right-click the panel and select the Add Drawer entry. If you want to add a whole menu of applications on the main menu to your panel, right-click the menu title displayed at the top of the menu and select the Add This As Drawer To Panel entry. The entire menu appears as a drawer on your panel, holding icons instead of menu entries (see Figure 8-11). For example, suppose you want to place the Internet applications menu on your panel. Right-click the Internet item and select Add This As Drawer To Panel. A drawer appears on your panel labeled "Internet", and clicking it displays a pop-up list of icons for all the Internet applications.

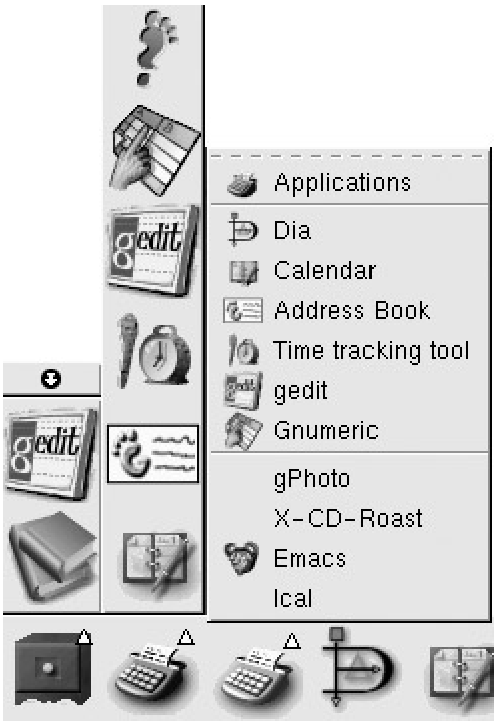


Figure 8-11: Gnome panel drawers and menus

A menu differs from a drawer in that a *drawer* holds application icons instead of menu entries. You can add menus to your panel, much as you add drawers. To add a submenu in the main menu to your panel, right-click the menu title and select the Add This As Menu To Panel entry. The menu title appears in the panel; you can click it to display the menu entries.

You can also add directory folders to a panel. Click and drag the Folder icon from the file manager window to your panel. Whenever you click this Folder button, a file manager window opens, displaying that directory. You already have a Folder button for your home directory. You can add directory folders to any drawer on your panel.

Moving and Removing Panel Objects

To move any object on the panel, even the Menu icon, just drag it with the middle mouse button. You can also right-click on it and choose Move Applet to let you move the applet. You can move it either to a different place on the same panel or to a different panel. When moving objects, you can have them either push over, switch with, or jump over other objects that they run into on the panel. To push over an object, you hold the SHIFT key down while moving the object. To switch with an object, you hold down the CTRL key; and to jump over an object, you hold down the ALT key.

To remove an object from the panel, right-click on it to display a pop-up menu for it and then and choose the Remove From Panel entry.

Main Menu

You open the main menu by clicking its button on the panel. The Main Menu button is a stylized picture of a bare foot. It is initially located on the left side of your panel, the lower left-hand corner of your screen. You only need to single-click the Main Menu button. You needn't keep holding your mouse button down. The menu pops up much like the Start menu in Windows.

You can configure menus using the Menu Properties dialog box. To change the properties for a menu on the panel, including the main menu, right-click its icon in the panel and select the Properties entry. This displays the Menu Properties dialog box, which has two sections: Menu Type and Main Menu. In the Main Menu section, you can set properties for that main menu. Several possible submenus can be displayed on the main menu, either directly or in other submenus. You can choose from the System, Applets, Favorites, KDE, and distribution menus. You can place these menus on the main menu or make them submenus. Distribution menus are those used for a specific distribution like Red Hat that are not specifically Gnome applications. KDE is used for KDE applications, if the KDE desktop is also installed on your system.

You can customize the main menu, adding your own entries, with the Menu Editor. To start the Menu Editor, select the Menu Editor entry in the Setting submenu located in the main menu. The Menu Editor is divided into two panes, the left being a tree view of the main menu. You can expand or shrink any of the submenus. The right pane holds configuration information for a selected entry. There are two panels: basic and advanced. The basic panel displays the name, command, and application type, as well as the icon. You can click the icon to change it. You can also change the Name, Command, or Type fields.

To add a new application, click the New Item button on the toolbar. The new item is placed in the currently selected menu. Enter the name, command, and type information, and then select an icon. Then click the Save button to add the entry to the menu. You can move the menu item in the menu by clicking the Up or Down Arrow button in the toolbar, or by dragging it with the mouse. If you are a user, remember you can only add entries to the User menu, not to the Systems menu.

An easier way to add an application is to use the drag-and-drop method. Locate the application you want to add with the file manager, and then drag and drop its icon to the appropriate menu in the Menu Editor. The entry is made automatically, using configuration information provided for that application by the file manager.

Panel Configuration

You use the Global Panel Configuration dialog box to configure properties for all Gnome panels. Either right-click the panel and select Global Properties or select Global Properties in the Panel submenu in the main menu to display this dialog window. The Global Panel Configuration dialog box has six tabbed panels: Animation, Launcher Icon, Drawer Icon, Menu Icon, Logout Icon, and Miscellaneous. With the Animation panel, you can enable panel animations, setting various options for them. The various icon panels enable you to select the images you want to use to denote active or inactive elements, among other features, such as border and depth. On the Miscellaneous panel, you set certain options such as allowing pop-up menus on the desktop, prompting before logout, or keeping panels below windows.

To configure individual panels, you use the Panel Properties dialog box. To display this dialog box, you right-click the particular panel and select the This Panel Properties entry in the pop-up menu, or select This Panel Properties in the main menu's Panel menu. For individual panels, you can set features for edge panel configuration and the background. The Panel Properties dialog box includes a tabbed panel for each. On the edge panel, you can choose options for positioning an edge panel and for minimizing it, including the Auto-Hide feature. The Hide Buttons feature enables you to hide the panel yourself.

On the Background panel, you can change the background image used for the panel. You can select an image, have it scaled to fit the panel, and select a background color. For an image, you can also drag and drop an image file from the file manager to the panel, and that image then becomes the background image for the panel.

Special Panel Objects

Special panel objects perform operations not supported by other panel objects. Currently these include the Lock, Logout, and Run buttons, as well as swallowed applications and the status dock. The Lock button shows a padlock and will lock your desktop, running the screensaver in its place. To access your desktop, click on it and then at the password prompt, enter your user password. The Logout button shows a monitor with a half- moon. Clicking it will display the Logout dialog box and you can then log out. It is the same as selecting Logout from the main menu. The Run button shows a hand on a terminal window. It opens the Run dialog box, which allows you to enter or select an application to run.

Any application can be run as applets on the panel. This process is referred to as swallowing the application. In effect, instead of being run in a Gnome window, the application is run as an applet on the panel. To swallow an application, you first start the application and take note of its window title. Then, in the Create Swallowed Application dialog box, enter the title of the application to swallow. You can further specify the applet's dimensions. If you want start the application directly as an applet, you can specify its program name in the Command window.

The status dock is designed to hold status docklets. A status docklet provides current status information on an application. KDE applications that support status docklets can use the Gnome status dock, when run under Gnome.

Gnome Applets

As previously stated, applets are small programs that perform tasks within the panel. To add an applet, right-click the panel and select Add to Panel, and then Applets from the pop-up menu. This, in turn, displays other pop-up menus listing categories of applets with further listings of available applets. Select the one you want. For example, to add the clock to your panel, select Clock from the Utility menu. To remove an applet, right-click it and select the Remove From Panel entry. You can also select the Applets menu in the Main menu and select and applet to add to your panel.

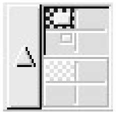
Gnome features a number of helpful applets, such as a CPU monitor and a mail checker. Some applets monitor your system, such as the Battery Monitor, which checks the battery in laptops; and CPU/MEM Usage, which shows a graph indicating your current CPU and memory use; as well as separate applets for CPU and memory load: CPULoad and MemLoad. The Mixer applet displays a small scroll bar for adjusting sound levels. The CD player displays a small CD interface for playing music CDs.

For network tasks, there are MailCheck, Modem Lights, and Web Control applets. MailCheck checks for received mail. To configure MailCheck, right-click it and select the Properties entry. You can set the frequency of checks, as well as specify a more sophisticated mail checker to run, such as **fetchmail**. The Modem Lights feature monitors your modem connection. You can configure it to monitor a PPP connection to an ISP over a modem. Web Control enables you to start your Web browser with a specified URL.

Several helpful utility applets provide added functionality to your desktop. The Clock applet can display time in a 12- or 24-hour format. Right-click the Clock applet and select the Properties entry to change its setup. You use the Printer applet to print your files. To print a file, drag its icon to the Printer applet. To configure the Printer applet, right- click it and select Properties. Here, you can specify the printer name and the printer command to use-helpful if you have more than one printer. The Drive Mount applet enables you to mount a drive using a single click. You can create a Drive Mount applet for each device you have, such as a floppy drive and a CD-ROM. To mount a file system, all you have to do is click the appropriate Drive Mount icon in the panel. To specify the file systems to mount, use the applet's Drive Mount Settings dialog box.

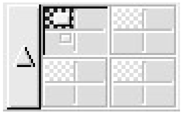
Gnome Desk Guide

The *Gnome Desk Guide*, shown next, appears in the panel and shows a view of your virtual desktops along with their desktop areas. Virtual desktops and their desktop areas are defined in the window manager. Desk Guide lets you easily move from one to another with the click of a mouse. The Gnome Desk Guide is a panel applet that works only in the panel.



Note If the Desk Guide is not already active, you can activate the Gnome Desk Guide by right-clicking the panel and selecting Add New Applet from the pop-up menu. This, in turn, displays other pop-up menus listing categories of applets and their listings of available applets. Select the Utility category and, in that menu, select Desk Guide.

The Desk Guide shows your entire virtual desktop as separate rectangles within a box. Each rectangle in turn is cut into small adjoining squares to show the desktop areas for each virtual desktop. Open windows show up as small colored rectangles in these squares. You can move any window from one virtual desktop or area to another by clicking and dragging its image in the Desk Guide with your middle mouse button. If you click the small arrow to the right of the Desktop view, the task list window opens, listing all the tasks (windows) currently open and running. (If the arrow is not displayed, open the Desk Guide properties and select Show Tacklist Arrow.) You can select a task to move to its window and the desktop it is currently positioned in. The previous illustration shows a simple Desk View applet displaying two virtual desktops, each with four desktop areas. The next illustration shows a more complex Desktop view with four virtual desktops each with four desktop areas.

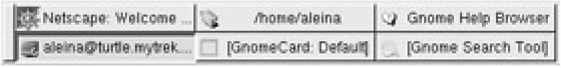


Note Various window managers use different terms to describe virtual desktops and their desktop areas. Enlightenment uses the terms "desktops" and "screens," whereas Sawfish uses "workspaces" that are then divided into "columns" and "rows." Desk Guide officially calls them "desktops" and "viewports."

To configure the Desk Guide, right-click it and select Properties to display the Properties dialog box. Here, you can choose from panels to configure the display, tasks, geometry, and advanced features like window manager options. You can set the size of the Desk Guide to extend beyond the height of the panel, elect to display any virtual desktop names, or even show hidden tasks. Remember, the window manager you are using may also have a pager you can use that may operate much like the Desk Guide. Check your window manager documentation on how to activate it.

Gnome Tasklist

The *Tasklist* shows currently opened applications. The Tasklist arranges tasks currently running in a series of buttons, one for each window. A task can be any open application, usually denoted by a window displayed on the screen. These can include applications such as a Web browser or a file manager window displaying a directory. You can move from one task to another by clicking on its button, in effect moving from one window to another. When you minimize a window, you can later restore it by clicking on its entry in the Tasklist. The next illustration shows a tasklist displaying buttons for several different kinds of windowed applications.



A right-click on a window's Tasklist entry opens a menu that lets you iconify or restore, shade, stick, or close the window. The iconify operation will reduce the window to its Tasklist entry. A right-click on the entry will display the menu with a Restore option instead of an iconify one, which you can then use to redisplay the window. The Shade entry will reduce the window to its title bar and the Stick entry will display the window in any desktop you move to. The Kill entry will close the window, ending its application.

To configure the Tasklist, you right-click on it and select the Properties entry. Here, you can set features such as the size of the tasklist, the number of rows, whether to display mini-icons, and the tasks to show.

Quicklaunch

You can use the Quicklaunch applet in the panel to start programs. The *Quicklaunch applet* holds a collection of small icons for application launchers. Click them to launch your application. The Quicklaunch applet can use only launchers that are already set up either on the main menu or on your desktop. To add a launcher to Quicklaunch, drag and drop the launcher to the Quicklaunch applet in the panel. A small icon is then created for it in the Quicklaunch applet. For main menu items, click an item and drag it to the Quicklaunch applet. Right-click a particular application's icon and select Properties to configure that launcher.

Gnome Configuration

You can configure different parts of your Gnome Interface using tools called *capplets.* Think of capplets as modules or plug-ins that can be added to enable you to configure various applications. Capplets exist for the core set of Gnome applications, as well as for other applications for which developers may have written capplets. To access capplets on Red Hat 7.2, you use the Start Here window and open the preferences window. This window will display icons for each category of capplets supported by the control center. Selecting one will open a window displaying icons for individual capplets. Double-clicking on a particular icon will open its caplet window. You can have several open at once (see Figure 8-12). You can also select a capplet from the Settings menu on the Gnome main menu.

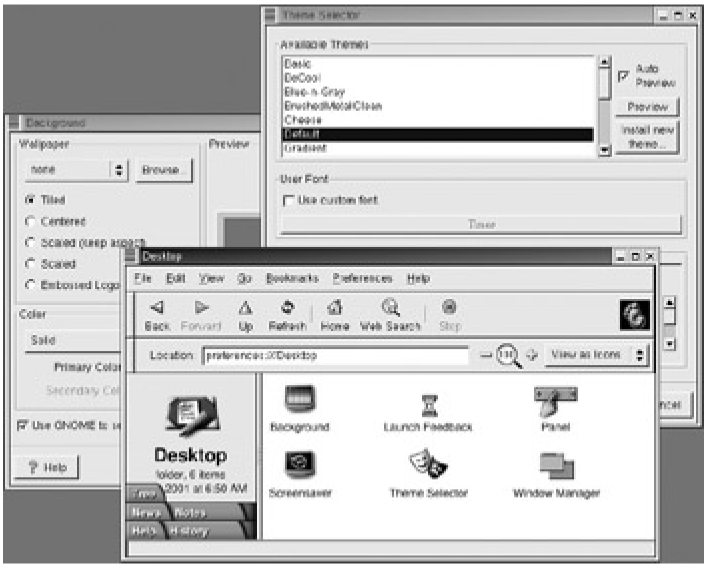


Figure 8-12: The Gnome Control Center Desktop capplets

Tip It is possible to also open the Control Center window listing all capplets using an expandable tree. To do so, you need to enter the **gnomecc** command in a terminal window.

Your Gnome system provides several desktop capplets you can use to configure your desktop: Background, Screensaver, Theme Selector, and Window Manager. You use the Background capplet to select a background color or image, the Screensaver capplet to select the screensaver images and wait time, the Theme selector to choose a theme, and the Window Manager capplet to choose the window manager you want to use.

On the Gnome Default Editor entry, you choose an editor as your default editor for Gnome, the editor the Gnome file manager uses to open text files. The Gnome File Types capplet enables you to specify Multipurpose Internet Mail Extensions (MIME) type entries for your system, associating given MIME types with certain applications. Notice that basic MIME type entries are already present. You can edit an entry and change its associated application. Also listed are Multimedia and Peripheral capplets. For the sound configuration, you can select sound files to play for events in different Gnome applications. For your keyboard, you can set the repeat sensitivity and click sound. You can configure mouse buttons for your right or left hand, and adjust the mouse motion. With the Session Manager capplet, you can configure certain Gnome session features, specifying non-Gnome programs to start up and whether you want a logout prompt.

Several User Interface capplets enable you to configure different interface components, such as menus, toolbars, and status bars. There are capplets for setting these features for applications, dialog boxes, and the Multiple Document Interface (MDI). You can specify whether toolbars and menus can be detached, whether they have relief borders, and whether they include icons. For dialog boxes, you can set features such as the arrangement of buttons or the position of the dialog box on the screen when it appears. The default MDI used for Gnome is modal. You can choose two other interfaces: toplevel and notebook.

Gnome sets up several configuration files and directories in your home directory. The **.gnome** directory holds configuration files for different desktop components, such as **gmc** for the file manager, **panel** for the panels, and **gmenu** for the main menu. **.gtkrc** holds configuration directives for the GTK+ widgets. The **.gnome-desktop** directory holds all the items you placed on your desktop.

Gnome Directories and Files

Most distributions install Gnome binaries in the **/usr/bin** directory on your system. Gnome libraries are located in the **/usr/lib** directory. Gnome also has its own **include** directories with header files for use in compiling and developing Gnome applications, **/usr/ include/libgnome** and **/usr/include/libgnomeui** (see Table 8-5). The directories located in **/usr/share/gnome** contain files used to configure your Gnome environment.

Table 8-5: Gnome Configuration Directories

System Gnome Directories	Contents
/usr/bin	Gnome programs
/usr/lib	Gnome libraries

Table 8-5: Gnome Configuration Directories

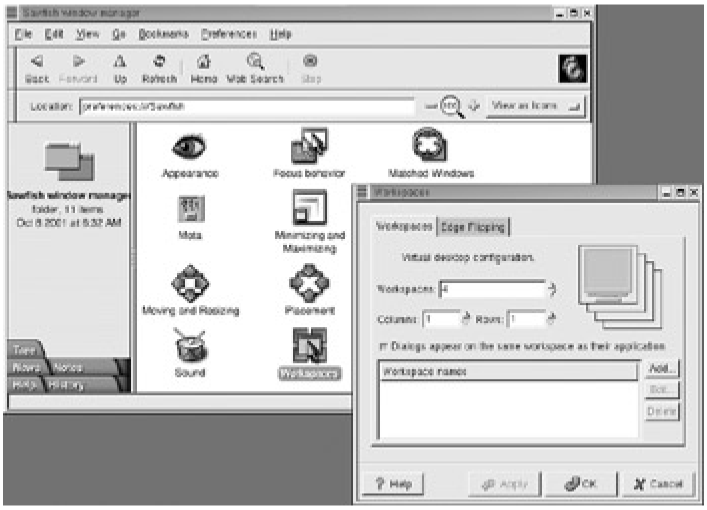
System Gnome Directories	**Contents**
/usr/include/libgnome	Header files for use in compiling and developing Gnome applications
/usr/include/libgnomeui	Header files for use in compiling and developing Gnome user interface components
/usr/share/gnome/apps	Files used by Gnome applications
/usr/share/gnome/help	Files used by Gnome Help system
/usr/share/doc/gnome*	Documentation for various Gnome packages, including libraries
/etc/X11/gdm/gnomerc	Gnome configuration file invoked with the Gnome Display Manager (GDM)
/etc/gconf	GConf configuration Files
User Gnome Directories	**Contents**
.gnome	Holds configuration files for the user's Gnome desktop and Gnome applications. Includes configuration files for the panel, Control Center, background, GnomeRPM, MIME types, and sessions
.gnome-desktop	Directory where files, directories, and links you place on the desktop will reside
.gnome-help-browser	Contains Gnome Help System configuration files, including history and bookmark files set up by the user
.gnome_private	The user private Gnome directory
.gtkrc	GTK+ configuration file
.gconf	GConf configuration database
.gconfd	GConf gconfd daemon management files
.nautilus	Configuration files for the Nautilus file manager (Gnome 1.4)

Gnome sets up several hidden directories for each user in their home directory that begin with **.gnome** and include a preceding period in the name. **.gnome** holds files used to configure a user's Gnome desktop and applications. Configuration files for the panel, Control Center, GnomeRPM, MIME types, and sessions, among others, are located here. The files **Gnome**, **GnomeHelp**, **Background**, and **Terminal** all hold Gnome configuration commands for how to display and use these components. For example, **Gnome** holds general display features for the desktop, while **GnomeHelp** specifies the history and bookmark files for the help system. Configuration files for particular Gnome applications are kept in the subdirectory **apps**. On Red Hat, the **redhat-apps** directory holds **.desktop** files containing Gnome instructions on how to handle different Red Hat utilities, such as netcfg. .gnome-desktop holds any files, folders, or links the user has dragged to the desktop. **.gnome-help-browser holds the bookmark and history files for the Gnome Help system. These are the bookmarks and** the list of previous documents the user consulted with the Gnome Help browser. **.gtckrc** is the user configuration file for the GTK+ libraries, which contains current desktop configuration directives for resources such as key bindings, colors, and window styles.

With Gnome 2.0, Gnome will officially implement GConf to provide underlying configuration support. GConf corresponds to the registry use on Windows systems. GConf consists of a series of libraries used to implement a configuration database for a Gnome desktop. This standardized configuration database allows for consistent interactions between Gnome applications. Gnome applications that are built from a variety of other programs, as Nautilus is, can use GConf to configure all those programs according to a single standard, maintaining configurations in a single database. Currently Red Hat implements the GConf database as XML files in the user's .gconf directory. Database interaction and access is carried out by the GConf daemon, gconfd.

Sawfish Window Manager

Sawfish is a fully Gnome-compliant window manager. It is designed to perform window managing tasks only and does not include features like application docks found on other window managers like Afterstep. This minimal approach to window management means that Sawfish does not duplicate Gnome desktop features as other window managers do (such as Enlightenment). At the same time, Sawfish is designed to be extensible, providing a Lisp-based set of commands you can use to fully configure all aspects of window management. Sawfish can be fully configured within Gnome using the Gnome Control Center sawfish capplet. First select Sawfish in Start Here's preferences window. This runs the Sawfish configuration program, displaying the Sawfish configuration window shown in Figure 8-13. You can find out more about Sawfish and obtain current versions from its Web site at **www.sawfish.org** or **sawmill.sourceforge.net** (Sawfish was originally named sawmill, so many of its Internet sites still bear that name). The site also provides detailed documentation of Sawfish commands and features. If you download the source code and want to compile it for use on Gnome, be sure to include the **--enable-capplet** option to add it in the Gnome Control Center.



Location: preferences:///Sawfish

Workspaces: 4

Columns: 1 Rows: 1

☑ Dialogs appear on the same workspace as their application

Figure 8-13: Sawfish configuration

The Sawfish Configuration window displays a list of configuration topics on the left and the panel for the selected topic on the right. Basic options set window displays, enabling you to select resize and move methods. With the Desktops option, you can create virtual desktops and specify the number of desktop areas for each one. On the panel are two configuration tools. The left one, labeled Size Of Virtual Screen, is used to determine the number of desktop areas. The right one, labeled Separate Desktops, is used to specify the number of virtual desktops. Recall, however, that the Gnome desktop is only supported on the first virtual desktop, not on any others. This means drag-and-drop operations do not work on the other virtual desktops. They do, however, work on any of the desktop areas on that first virtual desktop. The Gnome pager supports all the virtual desktops, displaying rectangles for each in the panel. Other topics cover features such as sounds, special effects, window focus, keyboard shortcuts, and backgrounds. You can set different backgrounds for each virtual desktop.

The Themes panel enables you to use a Sawfish theme, from which there are many to choose. Enlightenment is known for its magnificent themes. See **sawmill.themes.org** for themes you can download. To make a theme available, place it in your home directory's **.sawfish/themes** directory. Make sure that file has a **.sawfishtheme** extension. Sawfish maintains its own configuration directory, called **.sawfish**, in your home directory. It contains subdirectories for themes, backgrounds, and windows.

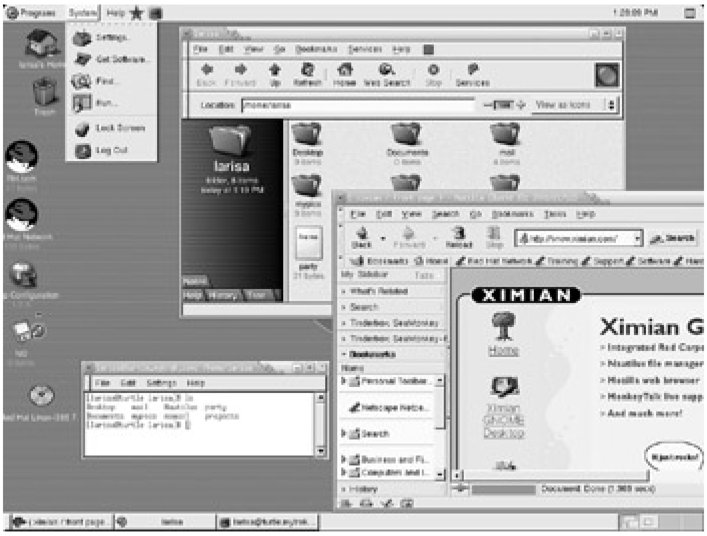
Gnome Themes

You can display your Gnome desktop using different themes that change the appearance of desktop objects such as windows, buttons, and scroll bars. Gnome functionality is not affected in any way. You can choose from a variety of themes. Many are posted on the Internet at **gtk.themes.org**. Technically, these are referred to as *GTK themes,* which allows the GTK widget set to change their look and feel.

To select a theme, use the Gnome Control Center and select Themes in the Desktop listing. You can select a theme from the Available Themes list on the Configuration panel. The Auto Preview button enables you to see an example of the theme. To use the theme, click Try. To install a theme you have downloaded from the Internet, click the Install New Theme button and locate the theme file. The theme is then installed on your system, and an entry for it appears in the Available Themes list.

The Ximian Gnome

Currently Ximian is developing software for Gnome, which is distributed under the GNU License, making it available to everyone free of charge. Ximian is an active member of the Gnome Foundation. Ximian is working to make the Gnome an effective and user-friendly desktop, adding enhancements such as improved desktop utilities and office applications. They currently provide an improved version of Gnome known as Ximian Gnome (see Figure 8-14) and are planning to offer a full suite of office applications. Currently, they offer a full-featured mail client called Evolution and a spreadsheet called Gnumeric. Evolution is a fully loaded communications application that includes a mail client, address book, calendar, and contact manager (still under development). Their next project is a full-featured word processor. You can obtain more information about Ximian from its Web site at **www.ximian.com**.



```
[larisa@turtle larisa]$ ls
Desktop    mail    Nautilus  party
Documents  mysics  nsmail    projects
[larisa@turtle larisa]$
```

Figure 8-14: Ximian Gnome

Ximian Gnome is an enhancement of Gnome that is fully compatible with all Gnome software. You first need to have Gnome installed on your system, and then you can download and install Ximian Gnome. You can download Ximian Gnome from its Web site at **www.ximian.com**, selecting the version for your particular Linux distribution.

Ximian Gnome displays a Menu panel at the top where you can access programs and perform tasks like logging out and locking your screen. A taskbar at the bottom of the screen shows currently running programs and windows. The Ximian Gnome interface is shown in Figure 8-14. However, there are several others that you can choose from. When you first run Ximian, you have the opportunity to customize your desktop, and you can choose from several arrangements, including a CDE panel or a traditional Gnome panel. In the Login window, you also have the option of logging into a traditional Gnome interface, labeled Gnome.

Updating Gnome

Currently, new versions of Gnome are being released frequently, sometimes every few months. Gnome releases are designed to enable users to upgrade their older versions easily. Be sure to obtain the release for your particular distributions (though you can install from the source code if you want). For Red Hat, you can use the Red Hat update agent to update any Gnome updates located on the Red Hat FTP site automatically. If you are using Ximian Gnome, you can use the Ximian red carpet update utility to automatically update Gnome.

Note The Gnome Web site (**www.gnome.org**) provides a link to Ximian at **www.ximian.com**, where you can download the Ximian version of Gnome for different distributions. Ximian versions tend to be more current.

Packages tailored for various distributions can also be downloaded from the Gnome FTP site at **ftp.gnome.org**. To manually download the update files and install them yourself, you first log in as the root user and then create a directory to hold the Gnome files. Then connect to an FTP site such as the Gnome FTP site at **ftp.gnome.org** or the **updates** directory for the Red Hat distributions located at **ftp.redhat.com**. You can use a Web browser, such as Netscape, but using an FTP client such as nctp, ftp, gFTP, or even the Gnome file manager (Nautilus or GMC) is preferable. Download the files for the new version to your new directory. For Red Hat, these are a series of RPM package files. To download using the Gnome file manager, enter the FTP URL in a file manager window's Location box to access the site. Move to the directory holding the Gnome files. Then select the files and drag and drop them to another file manager window that is open to the local directory in which you want them placed. The files are downloaded for you. To download using the **ftp client, be sure to turn off prompts with the prompt command and use mget *** to download all the files at once.

Once the RPM packages are downloaded, you can use the **rpm** command with the **-Uvh** option to install them or the GnomeRPM utility. Be sure to read any installation instructions first. These can be found in **readme** or **install** files. You may have to install some packages before others. The GnomeRPM utility will tell you if a certain package requires that other packages be installed first. It will list these other packages as dependencies, meaning that the package you want to install is dependent on them and needs to have these other packages already installed.

To install a particular Gnome RPM package manually, use the **rpm** command with the **-Uvh** options or an RPM package utility like GnomeRPM (see Chapter 31). This example installs the games package:

```
rpm -Uvh gnome-games-1.2.4-6.i386.rpm
```

Many of the most recent updates are provided in the form of source files that you can download and compile. These are usually packages in compressed archives with **.tar.gz** extensions. At **ftp.gnome.org,** these are currently located in **pub/Gnome/stable/ sources**. Check the Gnome Web site for announcements. For example, a new version of the Gnome core programs could be:

```
ftp.gnome.org/pub/GNOME/stable/sources/gnome-core/gnome-core-1.4.0.4.tar.gz
```

Once you download the archive, use the **tar** command with the **xvzf** options to decompress and extract it. In the directory generated, use the **./configure**, **make**, and **make** install commands to configure, create, and install the programs.

```
tar xvzf gnome-core-1.4.0.4.tar.gz
```

Chapter 9: The K Desktop Environment: KDE

Overview

The *K Desktop Environment* (*KDE*) is a network transparent desktop that includes the standard desktop features, such as a window manager and a file manager, as well as an extensive set of applications that cover most Linux tasks. KDE is an Internet-aware system that includes a full set of integrated network/Internet applications, including a mailer, a newsreader, and a Web browser. The file manager doubles as a Web and FTP client, enabling you to access Internet sites directly from your desktop. KDE aims to provide a level of desktop functionality and ease of use found in MAC/OS and Windows systems, combined with the power and flexibility of the Unix operating system.

Note KDE version 2.2 has superseded the earlier 1.1 version of KDE. This chapter describes version 2.2. There are many similarities with version 1.1; however, users familiar with the old KDE will find some important changes, such as a new file manager and control center.

The KDE desktop is developed and distributed by the KDE Project, which is a large open group of hundreds of programmers around the world. KDE is entirely free and open software provided under a GNU Public License and is available free of charge along with its source code. KDE development is managed by a core group: the KDE Core Team. Anyone can apply, though membership is based on merit.

Note KDE applications are developed using several supporting KDE technologies. These include KIO, which offers seamless and modular access of files and directories across a network. For interprocess communication, KDE uses the Desktop Communications Protocol (DCOP). KParts is the KDE component object model used to embed an application within another, such as a spreadsheet within a word processor. The XML GUI uses XML to generate and place GUI objects such as menus and toolbars. KHTML is an HTML 4.0 rendering and drawing engine.

Numerous applications written specifically for KDE are easily accessible from the desktop. These include editors, photo and paint image applications, spreadsheets, and office applications. Such applications usually have the letter *K* as part of their name-for example, KWord or KMail. A variety of tools are provided with the KDE desktop. These include calculators, console windows, notepads, and even software package managers. On a system administration level, KDE provides several tools for configuring your system. With KUser, you can manage user accounts, adding new ones or removing old ones. kppp enables you to connect easily to remote networks with Point-to-Point Protocol (PPP) protocols using a modem. Practically all your Linux tasks can be performed from the KDE desktop. KDE applications also feature a built-in Help application. Choosing the Contents entry in the Help menu starts the KDE Help viewer, which provides a Web page-like interface with links for navigating through the Help documents. KDE version 2.2 includes support for an office application suite called KOffice, based on KDE's KParts technology. KOffice includes a presentation application, a spreadsheet, an illustrator, and a word processor, among other components (see Chapter 14 for more details). In addition, an Interactive Development Environment (IDE), called KDevelop, is also available to help programmers create KDE-based software.

KDE was initiated by Matthias Ettrich in October 1996, and it has an extensive list of sponsors, including SuSE, Caldera, Red Hat, O'Reilly, DLD, Delix, Live, Linux Verband, and others. KDE is designed to run on any Unix implementation, including Linux, Solaris, HP-UX, and FreeBSD. The official KDE Web site is **www.kde.org**, which provides news updates, download links, and documentation. KDE software packages can be downloaded from the KDE FTP site at **ftp.kde.org** and its mirror sites. Several KDE mailing lists are available for users and developers, including announcements, administration, and other topics. See the KDE Web site to subscribe. A great many software applications are currently available for KDE at **apps.kde.com**. Development support and documentation can be obtained at **developer.kde.org**. Various KDE Web sites are listed in Table 9-1.

Table 9-1: KDE Web Sites

Web Site	Description
www.kde.org	KDE Web site
ftp.kde.org	KDE FTP site
apps.kde.com	KDE software repository
developer.kde.org	KDE developer site
www.trolltech.com	Site for Qt libraries
koffice.kde.org	KOffice office suite
kde.themes.org	KDE desktop themes
lists.kde.org	KDE mailing lists

Qt and Harmony

KDE uses as its library of GUI tools the Qt library, developed and supported by Troll Tech (**www.trolltech.com**). Qt is considered one of the best GUI libraries available for Unix/Linux systems. Using Qt has the advantage of relying on a commercially developed and supported GUI library. Also, using the Qt libraries drastically reduced the development time for KDE. Troll Tech provides the Qt libraries as open-source software that is freely distributable. Certain restrictions exist, however: Qt-based (KDE) applications must be free and open sourced, with no modifications made to the Qt libraries. If you develop an application with the Qt libraries and want to sell it, then you have to buy a license from Troll Tech. In other words, the Qt library is free for free applications, but not for commercial ones.

The Harmony Project is currently developing a free alternative to the Qt libraries. Harmony will include all Qt functionality, as well as added features such as multithreading and theming. It will be entirely compatible with any KDE applications developed using Qt libraries. Harmony will be provided under the GNU Library Public License (LGPL). See **www.gnu.org/software/harmony** for more information.

KDE Desktop

One of KDE's aims is to provide users with a consistent integrated desktop, where all applications use GUI interfaces (see Figure 9-1). To this end, KDE provides its own window manager (kwm), file manager (Konqueror), program manager, and desktop panel (Kicker). You can run any other X Window System-compliant application, such as Netscape, in KDE, as well as any Gnome application. In turn, you can also run any KDE application, including the Konqueror file manager, with any other Linux window manager, including Blackbox, Afterstep, and even Enlightenment. You can even run KDE applications in Gnome.

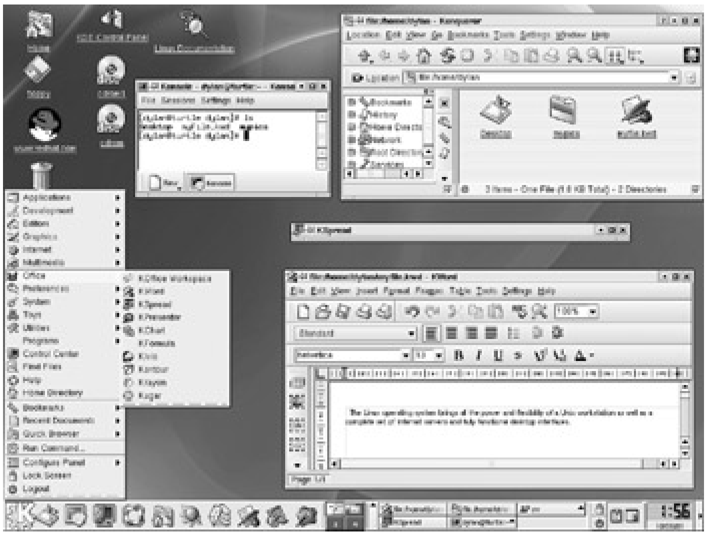


Figure 9-1: The KDE desktop

When you start KDE on Red Hat, the KDE panel is displayed at the bottom of the screen. Located on the panel are icons for menus and programs, as well as buttons for different desktop screens. The button for the K Menu shows a large *K* on a cog wheel with a small arrow at the top indicating it is a menu. This button is known as the Application Starter. Click this button to display the menu listing all the applications you can run. The K Menu operates somewhat like the Start menu in Windows. The standard KDE applications installed with the KDE can be accessed through this menu. You can find entries for different categories such as Internet, Systems, Multimedia, and Utilities. These submenus list KDE applications you can use. For example, to start the KDE mailer, select the Mail Client entry in the Internet submenu. To quit KDE, you can select the Logout entry in the K Menu. You can also right-click anywhere on the desktop and select the Logout entry from the pop-up menu. You can also click the Logout icon on the KDE panel located below the Lock icon. If you leave any KDE or X11 applications or windows open when you quit, they are automatically restored when you start up again. Should you just want to lock your desktop, you can click the Lock icon and your screen saver will appear. To access your desktop, click on the screen and a box appears prompting you for your login password. When you enter the password, your desktop reappears.

Note You can display a menu for desktop operations across the top of the desktop screen by selecting "Enable Desktop Menu" on the Desktop pop-up menu displayed when you right-click on the desktop. You can use this menu to create new shortcuts called desktop files for applications and devices, as well as for accessing open windows or changing to different virtual desktops. You can bring up the same set of menus by clicking anywhere on the desktop background.

A row of icons are displayed along the left side. These include a home directory folder icon, the Trash icon, a Printer icon, and floppy and CD-ROM icons. The Trash icon operates like the Recycle Bin in Windows or the trash can on the Mac. Drag items to it to hold them for deletion. To print a document you can drag it to the Printer icon. You can use the floppy and CD-ROM icons to mount, unmount, and display the contents of CD-ROMs and floppy disks.

Tip When you use KDE the first time, you are asked to personalize your desktop using KPersonalizer. To change your settings, you can run this program again by selecting Desktop Settings Wizard on the System menu. With KPersonalizer, you can select the kind of desktop style you want to use, changing the appearance of your windows and menus.

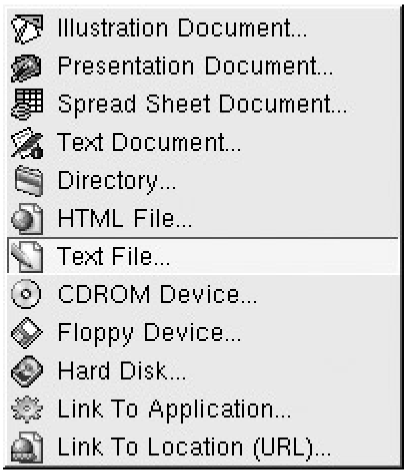
The KDE panel displayed across the bottom of the screen initially shows small buttons for the Application Starter, the window list, your home directory, the Help center, a terminal window, and buttons for virtual desktops, among others. The Window List icon looks like several grouped windows. It displays a list of all open windows and the desktop they are on. The Home Directory icon shows a folder with a house. Click it to open a file manager window showing your home directory. The Help Viewer icon is an image of a book. The Terminal Window icon is a picture of two computer monitors. Click this to open a terminal window where you can enter Linux shell commands.

The desktop supports drag-and-drop operations. For example, to print a document, drag it to the Printer icon. You can place any directories on the desktop by simply dragging them from a file manager window to the desktop. With KDE 2.2, the desktop also supports copy-and-paste operations, holding text you copied from one application in a desktop clipboard that you can then use to paste to another application. For example, you can copy a Web address from a Web page and then paste it into an e-mail message or a word processing document. This feature is supported by the Klipper utility located on the panel.

You can create new directories on the desktop by right-clicking anywhere on the desktop and selecting Create New and then Directory from the pop-up menu. All items that appear on the desktop are located in the **Desktop** directory in your home directory. There you can find the **Trash** directory, along with any others you place on the desktop. To configure your desktop, either click the Desktop icon located on the right of your panel or right-click the desktop and select the Display Properties entry. This displays a window with several tabbed panels for different desktop settings, such as the background or style.

Desktop Files

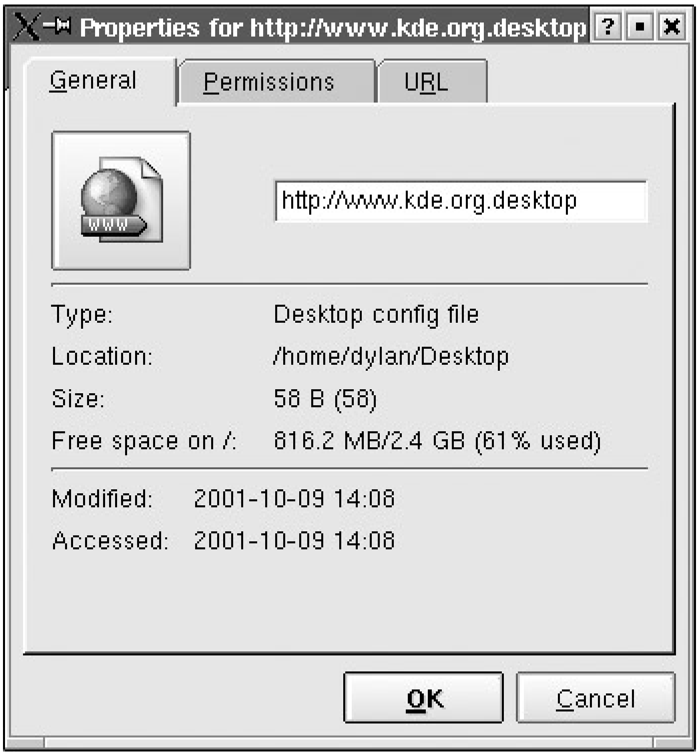
On the KDE 2.2 desktop, special files called *desktop* files are used to manage a variety of tasks, including device management, Internet connections, program management, and document types. You create a desktop file by right-clicking the desktop and then selecting Create New. From this menu, you choose the type of desktop file you want to create.



The Directory entry lets you create a link to a directory on your system. The CD-ROM and Floppy Device entries each create a desktop file that can mount CD-ROMs or floppy disks on your system. The Text File and HTML File entries are used to reference text files and Web pages on your system. The Link to Application entry is for launching applications. The Link to Location (URL) entry holds a URL address that you can use to access a Web or FTP site. The Application and Device entries are covered in the Applications section.

Given a desktop file that holds an Internet URL address, you can access that site directly by simply clicking the Desktop icon on the desktop. You can also place the desktop file on your desktop or put it in your panel, where it is easily accessible. You can configure the file to display any icon you choose. In effect, you can have an icon on your desktop you can click to immediately access a Web site. When you click the URL desktop file, the file manager starts up in its Web browser mode and will access that address, displaying the Web page. For FTP sites, it performs an anonymous login and displays the remote directory. The Red Hat icon labeled **www.redhat.com** in Figure 9-1 is an example of such a desktop file.

To create a URL desktop file, right-click the desktop and select the Create New menu, and then the Link to Location (URL) entry. A window appears that displays a box that prompts you to enter the URL name. Enter the URL address. You can later edit the desktop file by right-clicking on it and selecting Properties. A desktop dialog box for URL access is then displayed. This dialog box has three tabbed panels: General, Permissions, and URL (see Figure 9-2). On the General panel is the name of your desktop file. It will have as its name the URL address that you entered. You can change this to a more descriptive name if you wish. On the URL panel, you will see a box labeled URL with a URL you entered already in it. You can change it if you want. For example, for KDE themes, the URL would be **http://kde.themes.org**. Be sure to include the protocol, such as **http://** or **ftp://**. An Icon button on this panel shows the icon that will be displayed for this desktop file on your desktop. The default is a Web World icon. You can change it if you want by clicking the Icon button to open a window that lists icons you can choose from. Click OK when you are finished. The desktop file then appears on your desktop with that icon. Click it to access the Web site. An alternative and easier way to create a URL desktop file is simply to drag a URL from a Web page displayed on the file manager to your desktop. A desktop file is automatically generated with that URL. To change the default icon used, you can right-click the file and choose Properties to display the desktop dialog box. Click the Icon button to choose a new icon.



http://www.kde.org.desktop

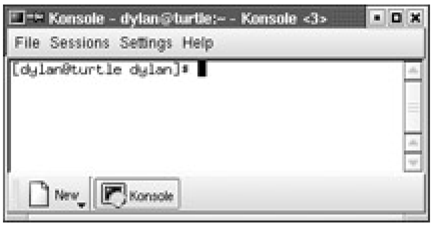
Type: Desktop config file
Location: /home/dylan/Desktop
Size: 58 B (58)
Free space on /: 816.2 MB/2.4 GB (61% used)

Modified: 2001-10-09 14:08
Accessed: 2001-10-09 14:08

Figure 9-2: The desktop dialog box

KDE Windows

A KDE window has the same functionality you find in other window managers and desktops. You can resize the window by clicking and dragging any of its corners or sides. A click-and-drag operation on a side extends the window in that dimension, whereas a corner extends both height and width at the same time. Notice that the corners are slightly enhanced. The top of the window has a title bar showing the name of the window, the program name in the case of applications, and the current directory name for the file manager windows. The active window has the title bar highlighted. To move the window, click this title bar and drag it where you want. Right-clicking the window title bar displays a drop-down menu with entries for window operations, such as closing or resizing the window. Within the window, menus, icons, and toolbars for the particular application are displayed. Here is an example of a KDE window.


Konsole - dylan@turtle:~ - Konsole <3>
File Sessions Settings Help
New
Konsole

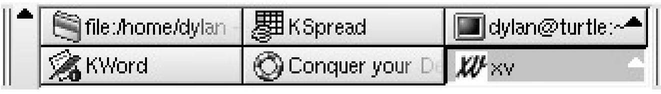
```
[dylan@turtle dylan]$
```

Opened windows are also shown as buttons on the KDE taskbar located on the panel. The taskbar shows the different programs you are running or windows you have open. This is essentially a docking mechanism that enables you to change to a window or application just by clicking its button. When you minimize (iconify) a window, it is reduced to its taskbar button. You can then restore the window by clicking its taskbar button.

KDE supports numerous different themes, each displaying window elements in different ways. The default KDE window theme for KDE 2.2 installed with Red Hat 7.2 is shown in these examples (the earlier version of KDE (1.0) uses a slightly different window theme). In the default KDE 2.2 window on Red Hat, there are two buttons to the left of the title bar at the top of the window. The leftmost button is used to display the window menu and shows an icon representing the type of window open. The button next to it is a Stick Pin button. The Stick Pin button is used to have a window appear on all your virtual desktops, no matter to which one you change. In effect, the window sticks on the screen when you change to another virtual desktop. When active, it appears as a stick pin pressed into the desktop. When inactive, it shows a stick pin on its side.

To the right of the title bar are three small buttons for iconifying, maximizing, or closing the window. The button with a square maximizes the window, letting the window take up the entire screen. Clicking the Maximize button with the middle or right mouse button maximizes vertically or horizontally. The rightmost button showing an *x* is used to close the window.

The button with the Dot icon is used to iconify the icon. When you click it, the window is no longer displayed on the desktop, but its button entry remains in the taskbar on the panel.



Click that button to redisplay the window. You can also reduce a window to its title bar by double-clicking the title bar. To restore the window, double-click the title bar again.

Application windows may also display a Help Notes button, shown next to the iconify button and displaying a question mark. Clicking this button changes your cursor to a question mark. You can then move the cursor to an item such as an icon on a toolbar, then click it to display a small help note explaining what the item does. Clicking a Forward button in the file manager taskbar will show a note explaining that this button performs a browser forward operation.

Tip When a window is not active, its contrast is reduced to make it easier for you to notice the active window.

As a multitasking operating system, Linux enables you to run several applications at the same time. This means you can have several applications open and running on your desktop, each with its own window. You can switch between them by moving from one window to another. When an application is open, a button for it is placed in the taskbar at the top of the desktop. You can switch to that application at any time by clicking its taskbar button. From the keyboard, you can use the ALT-TAB key combination to display a list of current applications. Holding down the ALT key and sequentially pressing TAB moves you through the list. You can hide an application at any time by clicking its window's Minimize button. The taskbar button entry for it remains. Click this to restore the application. Table 9-2 shows the KDE keyboard shortcuts.

Table 9-2: KDE Keyboard Shortcuts

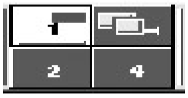
Keys	**Effect**
ALT-ESC or CTRL-ESC	Current session manager with Logout button
ALT-TAB and ALT-SHIFT-TAB	Traverse the windows of the current desktop
CTRL-TAB and CTRL-SHIFT-TAB	Traverse the virtual desktops

Table 9-2: KDE Keyboard Shortcuts

Keys	Effect
ALT-F2	Open small command line window
ALT-F3	Window operation menu
ALT-F4	Close window
CTRL-F[1…8]	Switch to a particular virtual desktop
CTRL-ALT-ESC	Force shutdown of X Windows

Virtual Desktops: The KDE Desktop Pager

KDE, like most Linux window managers, supports virtual desktops. In effect, this extends the desktop area on which you can work. You could have Netscape running on one desktop and be using a text editor in another. KDE can support up to 16 virtual desktops, though the default is four. Your virtual desktops can be displayed and accessed using the KDE Desktop Pager located on the panel. The KDE Desktop Pager represents your virtual desktops as miniature screens showing small squares for each desktop. By default there are four squares, numbered 1, 2, 3, and 4.



You can have up to 16. To move from one desktop to another, click the square for the destination desktop. Clicking 3 displays the third desktop, and clicking 1 moves you back to the first desktop.

Normally, when you open an application on a particular desktop, it appears only in that desktop. When you move to another desktop, the application disappears from your screen. Moving back again shows the application. For example, if you open KMail on the third desktop, and then move to the second desktop, KMail disappears from your screen. Moving back to the third desktop causes KMail to appear again. Selecting the taskbar button for an application also switches you to the desktop on which the application is open. In the example, clicking the KMail taskbar button switches to the third desktop. You can also use the Window list menu in the panel to display a listing of all open windows in each desktop. Selecting a window entry moves to that desktop. If you want an application to appear on all desktops, no matter which one you move to, click its window's Stick Pin button.

If you want to move a window to a different desktop, first open the window's menu by right-clicking the window's title bar. Then, select the To Desktop entry, which lists the available desktops. Choose the one you want. You can also right-click the window's title bar to display the window's menu.

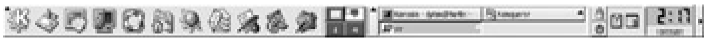
You can also configure KDE so that, if you move the mouse over the edge of a desktop screen, it automatically moves to the adjoining desktop. You need to imagine the desktops arranged in a four-square configuration, with two top desktops next to each other and two desktops below them. You enable this feature by selecting the Active Desktop Borders entry in the Desktop panel in the KDE Control Center.

To change the number of virtual desktops, you use the KPanel configuration window. From the K Menu, select Panel and then Configure. On the KPanel configuration window, select the Desktop panel. You then see entries for the current desktops. The visible bar controls the number of desktops. Slide this to the right to add more and to the left to reduce the number. The width bar controls the width of the desktop buttons on the panel. You can change any of the desktop names by clicking a name and entering a new one.

You can also configure desktop features, such as color background, for each virtual desktop. In the K Menu, select Settings and then Desktop. From this menu, you can choose various features to change. Selecting Background displays a Display Settings window. A list of virtual desktops is then shown. Select the one whose background you want to change, and then you can choose from colors and wallpaper. You can select wallpaper from a preselected list or choose your own.

KDE Panel: Kicker

The KDE panel (Kicker) is located at the bottom of the screen.



Through it, you can access most KDE functions. The panel includes icons for menus, directory windows, specific programs, and virtual desktops. At the left end of the panel is an icon with a large *K* on a cog wheel, known as the K button. This is the button for the KDE Application Starter that opens the K Menu. Click this button to display the menu of applications you run (you can also open the K Menu with the ALT-F1 key). From the KDE menu, you can access numerous submenus for different kinds of applications. The menu also includes certain key items such as Logout, to log out out of KDE; Lock Screen, to lock your desktop; Configure Panel, to access your Kicker panel configuration options; Run, to run programs from a command line; Quick Browser, to quickly browse your home, KDE, or root directories; and Recent Documents, which lists your recently opened documents.

To add an application to the panel, select the Add entry in the Configure Panel submenu located in the K Menu. You can also right-click anywhere on the panel and select Add from the pop-up menu. The Add menu displays the kind of objects you can add, such as buttons, applets, extensions, and windows. For KDE applications, select the Buttons entry. This lists all installed KDE applications. To add a button for an application to the panel, click the application entry. You can also drag applications from a file manager window or from the K menu, to the panel directly and have them automatically placed in the panel. The panel only displays desktop files. When you drag and drop a file to the panel, a desktop file for it is automatically generated. Kicker also supports numerous applets and several extensions. Applets are designed to run as icons in the panel. These include a clock, pager, and system monitor. Extensions add components to your desktop. For example, the Kasbar extension sets up its own panel and lists icons for each window you open. You can easily move from one window to another by clicking their corresponding icon in the Kasbar extension panel.

To configure the panel position and behavior, right-click the panel and select the Preferences entry. This displays a Panel Configuration dialog box with several tabbed panels: Position, Hiding, Look and Feel, Menus, Buttons, and Applets. The Position panel enables you to specify the edges of the screen where you want your panel and taskbar displayed. You can also enlarge or reduce it in size. On the Look and Feel panel, you can set a background theme.

The Menus panel lets you control the size of your menus as well as whether to display recently opened documents as menu items. On the Buttons page, you can have panel buttons display their background tiles, making each button more distinctive. The Applets panel lets you load only trusted applets or all available applets. Below each panel are buttons that you can use to restore the panel to its original settings (Default) or to its previous settings (Reset). You activate your settings either immediately with the Apply button or when you close the configuration window (OK).

You can add or remove menu items in your K Menu using the Edit Menus program. Right-click its *K* icon and select Configure. This launches the Edit Menus window, displaying two panes. The right pane shows the menu entries, and the left shows information about a selected entry. The right pane has two tabbed panels: General and Advanced. The General pane fields are where the features such as application program, the menu item name, and the icon used are specified. You can edit any of these entries. You can also specify if you want the program opened in a terminal window. On the Advanced panel you can select a keyboard shortcut for the program. To create a new menu entry, you select New in the File menu. You can also move, copy, and delete menu entries.

KDE Themes

For your desktop, you can select a variety of different themes. A *theme* changes the look and feel of your desktop, affecting the appearance of GUI elements, such as scroll bars, buttons, and icons. For example, you use the Mac OS theme to make your K Desktop look like a Macintosh. Themes for the K Desktop can be downloaded from the **kde.themes.org** Web site. Information and links for themes for different window managers can be found at **www.themes.org**. You can use the KthemeMgr program to install and change your themes.

The KDE Help System

The KDE Help viewer provides a browser-like interface for accessing and displaying both KDE Help files and Linux Man and info files. You can start the Help system either by selecting its entry in the K Menu, clicking on the Help icon in the Panel (life-preserver), or by right-clicking the desktop and selecting the Help entry (see Figure 9-3). The Help window is divided into two frames. The left frame of the Help screen holds two tabbed panels, one listing contents and the other providing a search engine. The left frame displays currently selected documents. A help tree on the content's panel lets you choose the kind of Help documents you want to access. Here you can choose manuals, Man pages, or info documents. The Help Center includes a detailed user manual, a FAQ, and KDE Web site access.

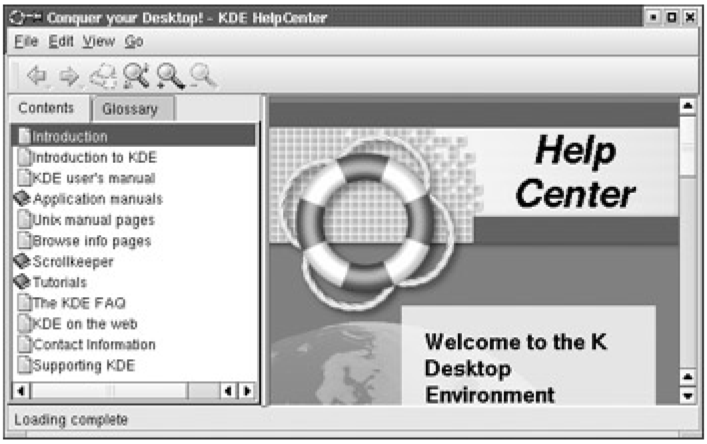


Figure 9-3: The KDE Help Center

You can also use a URL format to access Man and info pages, **info:** and **man:**. For example, **man:cp** displays the Man page for the **cp** command. A navigation toolbar enables you to move through previously viewed documents. KDE Help documents use an HTML format with links you can click to access other documents. The Back and Forward commands move you through the list of previously viewed documents. The KDE Help system provides an effective search tool for searching for patterns in Help documents, including Man and info pages. Select the Search entry to display a page where you can enter your pattern. You can also click the small icon in the toolbar of a page with a spyglass.

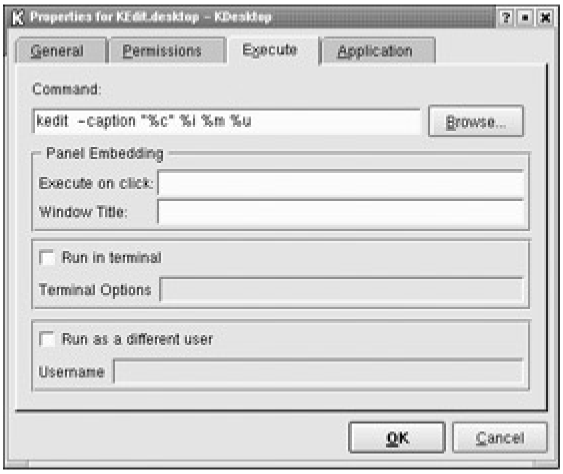
Applications

You can start an application in KDE in several ways. If an entry for it is in the K Menu, you can select that entry to start the application. Some applications also have buttons on the KDE panel you can click to start them. The panel already holds several of the commonly used programs, such as the Kate text editor and KMail. You can also use the file manager to locate a file using that application or the application program itself. Clicking its icon starts the application. Or, you can open a shell window and enter the name of the application at the shell prompt and press ENTER to start an application. You can also use the ALT-F2 key to open a small window consisting of a box to enter a single command. You can use the UP ARROW and DOWN ARROW keys to display previous commands, and the RIGHT ARROW and LEFT ARROW keys or the BACKSPACE key to edit any of them. Press ENTER to execute a command.

Note You can create a desktop file on your desktop for any application already on your KDE menu by simply clicking and dragging its menu entry to the desktop. Select Copy and a desktop file for that application is created for you on your desktop, showing its icon.

You can also access applications directly from your desktop. To access an application from the desktop, either create a desktop file or a standard link file that can link to the original application program. With a desktop file, you can choose your own icon and specify a tooltip comment. You can also use a desktop file to start a shell-based application running in its own terminal window. A standard link, on the other hand, is a simple reference to the original program file. Using a link starts the program up directly with no arguments. To create a standard link file, locate the application on your file system, usually in the **/bin**, **/usr/bin**, or **/usr/sbin** directories. Then, click and drag the application icon to your desktop. In the pop-up menu, select Link. The link has the same icon as the original application. Whenever you then click that icon, you can select Start from the pop-up menu to start the application. You can also use this method to run an application program you have created yourself, locating it in your own directory and creating a link for it on your desktop.

To create a new desktop file for an application, right-click anywhere on the empty desktop, select Create New from the pop-up menu, and then choose "Link to application." Enter the name for the program and a desktop file for it appears on the desktop with that name. A Properties dialog box then opens with four panels: General, Permissions, Execute, and Application. The General panel displays the name of the link. To specify the application the desktop file runs, go to the Execute panel and either enter the application's program name in the Execute box or click Browse to select it (see Figure 9-4). If this is a shell program, you can elect to run it from within a terminal window. To select an icon image for the desktop file, click the Cog icon. The Select Icon window is displayed, listing icons from which you can choose. To run a shell-based program such as Pine or Vi, click the "Run in terminal" check box and specify any terminal options. Certain KDE programs can minimize to a small icon, which can be displayed in the panel while they are running. This is referred to as *swallowing on the panel.* Enter the name of the program in the Execute box.



Command:
kedit -caption "%c" %i %m %u

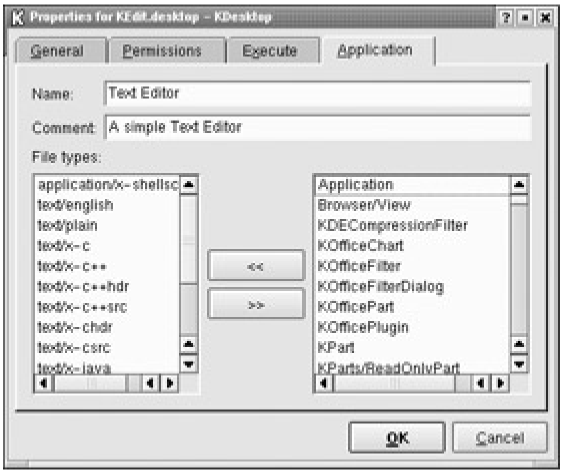
Panel Embedding
Execute on click:
Window Title:

□ Run in terminal
Terminal Options

□ Run as a different user
Username

Figure 9-4: KDE Application Desktop dialog box

On the Permissions panel, be sure to set execute permissions so that the program can be run. You can set permissions for yourself, for your group, or for any user on the system. In the Application panel, you can specify the type of documents to be associated with this application. The bottom of the panel shows two lists. The left list is for MIME types you want associated with this program, and the right list is the listing of available MIME types from which to choose (see Figure 9-5). To add a MIME type, select an entry in the right list and click the Left Arrow button. Use the Right Arrow button to remove a MIME type. On the panel, you also specify the comment, the file manager program name, and the name in your language. The comment is the Help note that appears when you pass your mouse over the icon. For the file manager program name, enter the name followed by a semicolon. This is the name used for the link, if you use the file manager to display it. Desktop files needn't reside on the desktop. You can place them in any directory and access them through the file manager. You can later make changes to a desktop file by right-clicking its icon and selecting Properties from the pop-up menu. This again displays the dialog box for this file. You can change its icon and even the application it runs. You can download other icons from **icons.themes.org**.



Name: Text Editor

Comment: A simple Text Editor

File types:

application/x-shellsc
text/english
text/plain
text/x-c
text/x-c++
text/x-c++hdr
text/x-c++src
text/x-chdr
text/x-csrc
text/x-java

Application
Browser/View
KDECompressionFilter
KOfficeChart
KOfficeFilter
KOfficeFilterDialog
KOfficePart
KOfficePlugin
KPart
KParts/ReadOnlyPart

Figure 9-5: KDE Application Desktop MIME type entries

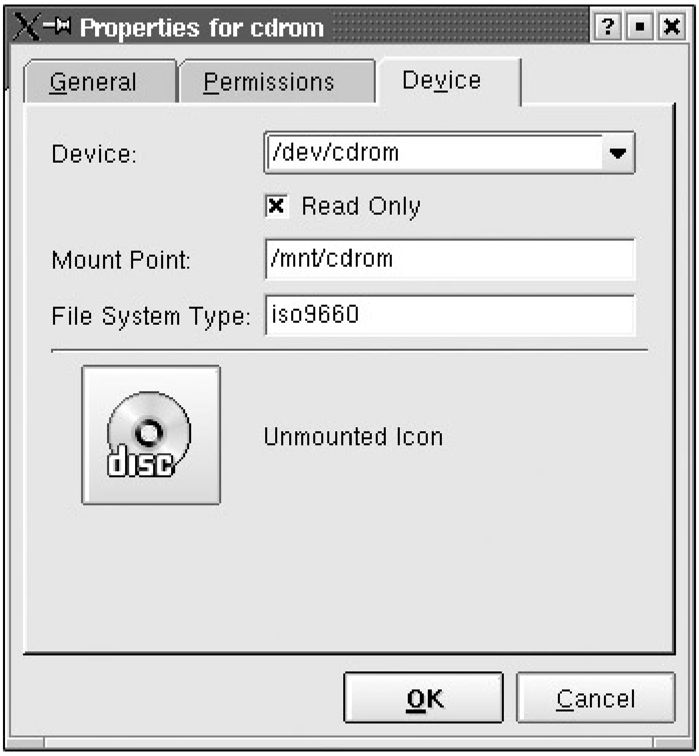
You can have KDE automatically display selected directories or start certain applications whenever it starts up. To do so, place links for these windows and applications in the **AutoStart** directory located in your **.kde** directory. To place a link for a directory in the AutoStart folder, first locate the Directory icon using the file manager. Then, click and drag the icon to the AutoStart folder. From the pop-up menu that appears, select Link (do not select Copy or Move). Whenever you start KDE, that directory is displayed in a file manager window. You can do the same for applications and files. Locate the application with the file manager and click and drag it to the AutoStart folder, selecting Link. For a file, do the same. Whenever KDE starts, those applications automatically start. For files, the application associated with a file starts using that file. For example, to start KMail automatically, click and drag its icon to the AutoStart folder, selecting Link.

Mounting CD-ROMs and Floppy Disks from the Desktop

Red Hat created desktop icons for your CD-ROMs and floppies when it installed KDE. Floppy and CD-ROM icons are displayed on the left side of your KDE desktop. To access a CD-ROM disk, place the CD-ROM disk in your CD-ROM drive and click the CD-ROM icon. The file manager window then opens, displaying the contents of the CD-ROM's top-level directory. You can also right-click the icon to display a pop-up menu with an entry to mount or unmount the disk. When the CD-ROM holds a mounted CD disk, the CD-ROM icon displays a small red rectangle on its image. Unlike on Windows systems, the CD-ROM disk remains locked in the CD-ROM drive until you unlock it. To unmount the CD, right-click the CD-ROM's icon and select Unmount from the pop-up menu. You can then open the CD-ROM drive and remove the CD.

To access a floppy disk, you can perform a similar operation using the Floppy Disk icon. Place the floppy disk in the disk drive and click the Floppy Disk icon. This displays a file manager window with the contents for the floppy disk. Or, you can right-click the icon to display a pop-up menu with an entry to mount the disk. Once it is mounted, you can access it, copying files to and from the disk. Be careful not to remove the disk unless you first unmount it. To unmount the disk, right-click its icon and select Unmount from the icon's pop-up menu. You can perform one added operation with floppy disks. If you put in a blank disk, you can format it. You can choose from several file system formats, including MS-DOS. To format a standard Linux file system, select the ext2 entry.

A desktop file you use for your CD-ROM is a special kind of desktop file designed for file system devices. Should you add a new CD-ROM or floppy drive, you can create a new desktop file for it to enable you to access the drive from your desktop. To create one, first right-click anywhere on the desktop, select New, and then select either CD-ROM Device for a CD-ROM drive or Floppy Device for a floppy drive. This opens a Properties window with tabs for General, Permissions, and Device. In the General tab you can set the name for the device icon that will appear on the desktop as well as choose the icon you want to show for a mounted CD-ROM or floppy disk (a default is already provided). On the Device panel, you select the actual device, its mount point on your file system, and the type of file system it will mount, as well as the icon used to indicate when it is unmounted (see Figure 9-6). On the Permissions panel, you can also indicate the permissions that have been set to allow access to the device. See the chapter on file administration, Chapter 32, for a discussion on devices and file systems. The desktop file does not perform the necessary system administration operations that enable access to the CD-ROM by ordinary users. Normally, only the systems administrator (root user) can mount or unmount CD-ROMs and floppy disks. You also must make sure an entry is in the **/etc/fstab** file for the CD-ROM or floppy drive. If not, you have to add one. Such operations are fairly easy to perform using the file system management tools provided by Linuxconf and fsconf. Check Chapter 4 and Chapter 32 for the procedures to use.



Device: /dev/cdrom

☒ Read Only

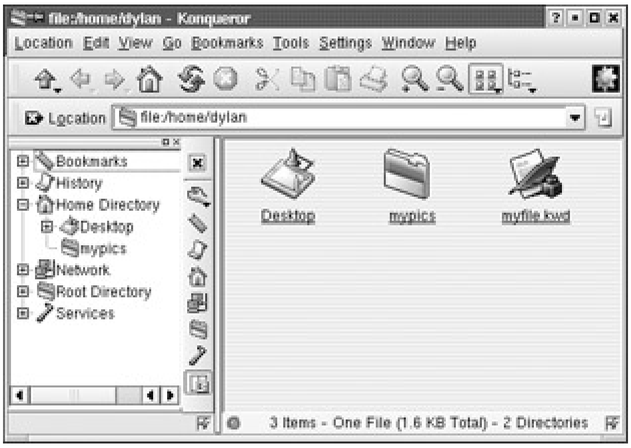
Mount Point: /mnt/cdrom

File System Type: iso9660

Figure 9-6: The Desktop dialog box for CD-ROM devices

KDE File Manager and Internet Client: Konqueror

The KDE file manager is a multifunctional utility with which you can manage files, start programs, browse the Web, and download files from remote sites (see Figure 9-7). Traditionally, the term "file manager" was used to refer to managing files on a local hard disk. The KDE file manager extends its functionality well beyond this traditional function because it is Internet capable, seamlessly displaying remote file systems as if they were your own, as well as viewing Web pages with browser capabilities. It is capable of displaying a multitude of different kinds of files, including image, postscript, and text files. KOffice applications can be run within the Konqueror window. With KDE 2.2, the original KDE file manager, kfm, was replaced by a new file manager called Konqueror.



Location file:/home/dylan

3 Items - One File (1.6 KB Total) - 2 Directories

Figure 9-7: The KDE file manager

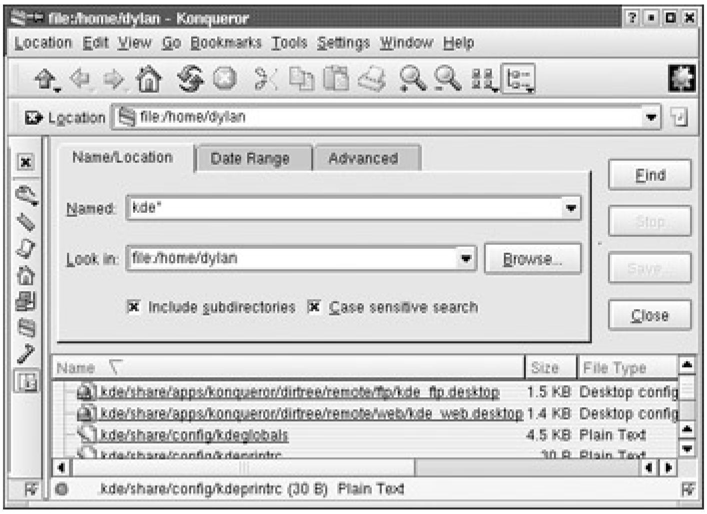
A KDE file manager window consists of a menu bar, a navigation toolbar, a location field, a status bar, and a sidebar that provides different views of user resources, such as a tree view of file and directory icons for your home directory. When you first display the file manager window, it displays the file and subdirectory icons for your home directory. Files and directories are automatically refreshed. So, if you add or remove directories, you do not have to manually refresh the file manager window. It automatically updates for you, showing added files or eliminating deleted ones. The files listed in a directory can be viewed in several different ways. You can view files and directories as icons, small icons, or in a detailed listing. The detailed listing provides permissions, owner, group, and size information. Permissions are the permissions controlling access to this file (see Chapter 12). Configuration files are not usually displayed. These are files beginning with a period and are often referred to as *dot files.* To have the file manager display these files, select Show Dot Files from the View menu.

The sidebar lists different resources that a user can access with Konqueror. The sidebar has both a classic and extended version. You can select which one to use from the Window menu. In the classic version, resources such as file manager history, bookmarks, and your home directory are listed in an expandable tree. Click an entry to expand it. Double-click to access it with Konqueror. For example, to move to a subdirectory, expand your home directory entry and then double-click the subdirectory you want. Konqueror will now display that subdirectory. To go to a previously bookmarked directory or Web page, find its entry in the Bookmarks listing and select it.

The extended sidebar features a vertical button bar for displaying items such as your file manager history, home directory, bookmarks, and network resources. The History button lists the network resources and directories you have accessed, including Web pages. The Network button will list network resources you have access to, such as FTP and Web sites. The Folder button will display your system's root directory. If Multiple Views are enabled, you can display several of these at once, just by clicking the ones you want. If Multiple Views are not enabled, then the previous listing is replaced by the selected one. Turn off a display by clicking its button again. The last button is a Classic Sidebar button which will display all of the resources in an expandable tree, like the classic sidebar does.

To configure the extended sidebar, click on its Configure button in the Sidebar Button bar. Select the multiple views entry to allow the display of several resource listings at once, each in their separate sub-sidebar. You can also add a new resource listing, choosing from a bookmark, history, or directory type. A button will appear for the new listing. You can right-click the button to select a new icon for it or select a URL, either a directory pathname or a network address. To remove a button and its listing, right-click on it and select the Remove entry.

To search for files, select the Find entry in the Tools menu or click on the Looking Glass icon. This opens a pane within the file manager window in which you can search for filenames using wildcard matching symbols, such as *. Click the Find button to run the search and on the Stop button to stop it (see Figure 9-8). The search results are displayed in a pane in the lower half of the file manager window. You can click a file and have it open with its appropriate application. Text files are displayed by the Kate text editor. Images are displayed by KView, and postscript files by KGhostview. Applications are run. The search program also enables you to save your search results for later reference. You can even select files from the search and add them to an archive.


file:/home/dylan - Konqueror
Location Edit View Go Bookmarks Tools Settings Window Help
Name/Location
Date Range
Advanced
Find
Stop
Save...
Close
Browse...

Location: file:/home/dylan

Named: kde*

Look in: file:/home/dylan

☒ Include subdirectories ☒ Case sensitive search

Name	Size	File Type
kde/share/apps/konqueror/dirtree/remote/ftp/kde_ftp.desktop	1.5 KB	Desktop config
kde/share/apps/konqueror/dirtree/remote/web/kde_web.desktop	1.4 KB	Desktop config
kde/share/config/kdeglobals	4.5 KB	Plain Text
kde/share/config/kdeprintrc	30 B	Plain Text

kde/share/config/kdeprintrc (30 B) Plain Text

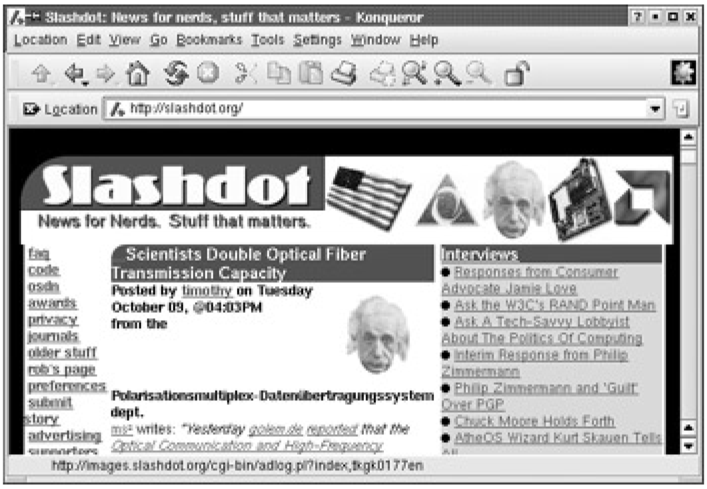
Figure 9-8: The KDE Search tool

You can open a file either by clicking it or by selecting it, and then choosing the Open entry in the File menu. A single-click, not a double-click, opens the file. If you want to select the file or directory, you need to hold down the CTRL key while you click it. A selection is performed with a CTRL-click. If the file is a program, that program starts up. If it is a data file, such as a text file, then the associated application is run using that data file. For example, if you click a text file, the Kate application starts displaying that file. If Konqueror cannot determine the application to use, it opens a dialog box prompting you to enter the application name. You can click the Browse button on this box to use a directory tree to locate the application program you want.

The file manager can also extract tar archives and install RPM packages. An *archive* is a file ending in **.tar.gz**, **.tar**, or **.tgz**. Clicking the archive lists the files in it. You can extract a particular file simply by dragging it out the window. Clicking a text file in the archive displays it with Kate, while clicking an image file displays it with KView. Selecting an RPM package opens it with the kpackage utility, which you can then use to install the package.

Moving Through the File System

A single-click on a directory icon moves to that directory and displays its file and subdirectory icons. Unlike other interfaces, KDE does not use double-clicking to open a directory. To move back up to the parent directory, you click the Up Arrow button located on the left end of the navigation toolbar. A single-click on a directory icon moves you down the directory tree, one directory at a time. By clicking the Up Arrow button, you move up the tree. To move directly to a specific directory, you can enter its pathname in the Location box located just above the pane that displays the file and directory icons. Figure 9-9 shows the KDE file manager window displaying the current directory. You can also use several keyboard shortcuts to perform such operations, as listed in Table 9-3.


Slashdot: News for nerds, stuff that matters - Konqueror
Location Edit View Go Bookmarks Tools Settings Window Help
Location
http://slashdot.org/
Slashdot
News for Nerds. Stuff that matters.
faq
code
osdn
awards
privacy
journals
older stuff
rob's page
preferences
submit
story
advertising
supporters
Interviews
Responses from Consumer Advocate Jamie Love
Ask the W3C's RAND Point Man
Ask A Tech-Savvy Lobbyist About The Politics Of Computing
Interim Response from Philip Zimmermann
Philip Zimmermann and 'Guilt' Over PGP
Chuck Moore Holds Forth
AtheOS Wizard Kurt Skauen Tells All
http://images.slashdot.org/cgi-bin/adlog.pl?index,tkgk0177en

Scientists Double Optical Fiber Transmission Capacity

Posted by timothy on Tuesday October 09, @04:03PM from the Polarisationsmultiplex-Datenübertragungssystem dept.

ms² writes: "Yesterday golem.de reported that the Optical Communication and High-Frequency

Figure 9-9: File manager as Web browser

Table 9-3: KDE File Manager Keyboard Shortcuts

Keys	Description
ALT-LEFT ARROW	Back in History
ALT-RIGHT ARROW	Forward in History
ALT-UP ARROW	One directory up
ENTER	Open a file/directory
ESC	Open a pop-up menu for the current file
LEFT/RIGHT/UP/DOWN ARROWS	Move among the icons
SPACEBAR	Select/unselect file
PAGE UP	Scroll up fast
PAGE DOWN	Scroll down fast
RIGHT ARROW	Scroll right (on Web pages)
LEFT ARROW	Scroll left (on Web pages)
UP ARROW	Scroll up (on Web pages)
DOWN ARROW	Scroll down (on Web pages)
CTRL-C	Copy selected file to Clipboard
CTRL-V	Paste files from Clipboard to current directory
CTRL-S	Select files by pattern

Table 9-3: KDE File Manager Keyboard Shortcuts

Keys	**Description**
CTRL-T	Open a terminal in the current directory
CTRL-L	Open new location
CTRL-F	Find files
CTRL-W	Close window

Like a Web browser, the file manager remembers the previous directories it has displayed. You can use the Back and Forward Arrow buttons to move through this list of prior directories. For example, a user could use the Location field to move to the **~/birthday** directory and use it again to move to the **~/reports** directory. Clicking the Back Arrow button displays the ~/**birthday** directory (the ~ represents the user's home directory). Then, clicking the Forward Arrow button moves it back to the **~/reports** directory. You can move directly to your home directory by clicking the Home button. This has the same effect as the **cd** command in the shell. If you know you want to access particular directories again, you can bookmark them, much as you do a Web page. Just open the directory and select the Add Bookmarks entry in the Bookmark menu. An entry for that directory is then placed in the file manager's Bookmark menu. To move to the directory again, select its entry in the Bookmark menu. This is helpful for directories you might use frequently or for directories you must access with lengthy or complex pathnames. Bookmarks also apply to individual files and applications. You can even bookmark desktop icons. To bookmark a file, first select a file and then choose the Add Bookmarks entry in the Bookmark menu. Later, selecting that bookmark opens that file. You can do the same thing with applications, where selecting the application's bookmark starts the application. Each bookmark is a file placed in your **.kde2/share/apps/konqueror/bookmarks** directory (kfm instead of Konqueror if you are upgrading from KDE 1.0). You can go to this directory and change the names of any of the files, and they then appear as changed on your Bookmark menu. To change their names, right-click the file and select Properties from the pop-up menu. In the dialog box displayed, you see the filename is the full pathname on the General tab. You can replace the pathname with one of your own choosing. This bookmark would then appear as **myreport**. The pathname used to access the file is actually on the URL panel.

To help you navigate from one directory to another, you can use the Location field or the directory tree. In the Location field, you can enter the pathname of a directory, if you know it, and press ENTER. The file manager then displays that directory. The directory tree provides a tree listing all directories on your system and in your home directory. You can activate the directory tree by selecting the Show Tree entry in the View menu. The directory tree has three main entries: Root, My Home, and Desktop. The Root entry displays the directories starting from the system root directory, the My Home entry displays directories starting from your home directory, and the Desktop entry displays the files and links on your desktop. Click a side triangle to expand a directory entry, and click a down triangle of an expanded directory entry to hide it.

Internet Access

The KDE file manager doubles as a Web browser and an FTP client. It includes a box for entering either a pathname for a local file or a URL for a Web page on the Internet or your intranet. A navigation toolbar can be used to display previous Web pages or previous directories. The Home button will always return you to your home directory. When accessing a Web page, the page is displayed as on any Web browser. With the navigation toolbar, you can move back and forth through the list of previously displayed pages in that session. This feature is particularly convenient for displaying local Web pages, such as documentation in HTML format. Most Linux distributions provide extensive documentation in the form of Web pages you can easily access and display using a KDE file manager window. Figure 9-9, shown previously, shows the KDE file manager window, operating as a Web browser and displaying a Web page.

The KDE file manager also operates as an FTP client. When you access an FTP site, you navigate the remote directories as you would your own. The operations to download a file are the same as copying a file on your local system. Just select the file's icon or entry in the file manager window and drag it to a window showing the local directory to which you want it downloaded. Then, select the Copy entry from the pop-up menu that appears.

By default, KDE attempts an anonymous login. If you want to perform a nonanonymous login as a particular user, add the user name with an @ symbol before the FTP address. You are then prompted for the user password. For example, the following entry logs in to the **ftp.mygames.com** server as the user **chris**:

```
ftp://chris@ftp.mygames.com
```

Copy, Move, Delete, and Archive Operations

To perform an operation on a file or directory, you first have to select it. In KDE, to select a file or directory, you hold the CTRL key down, while clicking the file's icon or listing. To select more than one file, continually hold the CTRL key down while you click the files you want. Or, you can use the keyboard arrow keys to move from one file icon to another and then use the SPACEBAR to select the file you want.

To copy and move files, you can use the standard drag-and-drop method with your mouse. To copy a file, you locate it by using the file manager. Open another file manager window to the directory to which you want the file copied. Then, click and drag the File icon to that window (be sure to keep holding down the mouse button). A pop-up menu appears with selections for Move, Copy, or Link. Choose Copy. To move a file to another directory, follow the same procedure, but select Move from the pop-up menu. To copy or move a directory, use the same procedure as for files. All the directory's files and subdirectories are also copied or moved.

You can also move or copy files using the Copy and Paste commands. First, use a CTRL-click to select a file or directory (hold the CTRL key down while clicking the File icon). Either select the Copy entry from the Edit menu or click the Copy button in the navigation bar. Change to the directory to which you want to copy the selected file. Then, either select Paste from the Edit menu or click the Paste button. Follow the same procedure for moving files, using the Move entry from the Edit menu or the Move button in the navigation toolbar. Most of the basic file manager operations can be selected from a pop-up menu displayed whenever you right-click a file or directory. Here, you can find entries for copying, moving, and deleting the file, as well as navigating to a different directory.

Distinguishing between a copy of a file or directory and a link is important. A *copy* creates a duplicate, whereas a *link* is just another name for the same item. Links are used extensively as ways of providing different access points to the same document. If you want to access a file using an icon on your desktop, creating a link on the desktop (rather than a copy) is best. With a copy, you have two different documents, whereas with a link you are accessing and changing the same document. To create a link on your desktop, click and drag the File icon from the directory window to the desktop and select Link from the pop-up menu. Clicking the link brings up the original document. Deleting the link only removes the link, not the original document. You can create links for directories or applications using the same procedure. You can also place links in a directory: click and drag the files to which you want to link into that directory and select Link from the pop-up menu.

You delete a file by removing it immediately or placing it in a Trash folder to delete later. To delete a file, select it and then choose the Delete entry in the Edit menu. You can also right-click the icon and select Delete. To place a file in the Trash folder, click and drag it to the Trash icon on your desktop or select Move to Trash from the Edit menu. You can later open the Trash folder and delete the files. To delete all the files in the Trash folder, right-click the Trash icon and select Empty Trash Bin from the pop-up menu. To restore any files in the Trash bin, open the Trash bin and drag them out of the Trash folder.

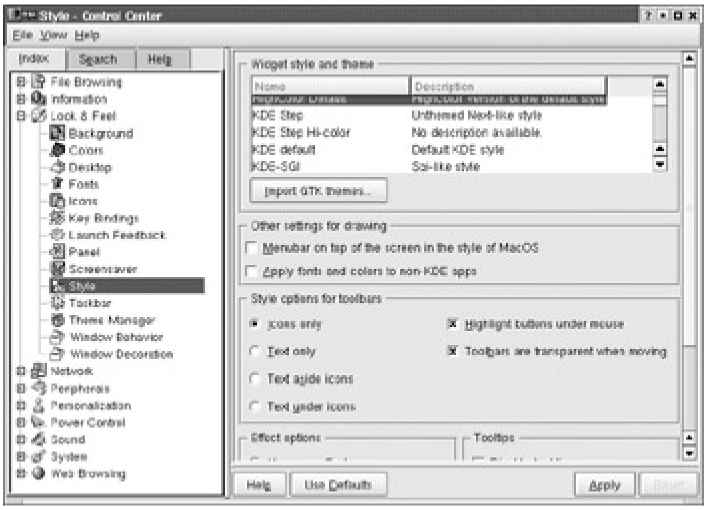
Each file or directory has properties associated with it that include permissions, the filename, and its directory. To display the Properties window for a given file, right-click the file's icon and select the Properties entry. On the General panel, you see the name of the file displayed. To change the file's name, replace the name there with a new one. Permissions are set on the Permissions panel. Here, you can set read, write, and execute permissions for user, group, or other access to the file. See Chapter 12 for a discussion of permissions. The Group entry enables you to change the group for a file.

.directory

KDE automatically searches for and reads an existing **.directory** file located in a directory. A **.directory** file holds KDE configuration information used to determine how the directory is displayed. You can create such a file in a directory and place a setting in it to set display features, such as the icon to use to display the directory folder.

KDE Configuration: KDE Control Center

With the KDE Control Center, you can configure your desktop and system, changing the way it is displayed and the features it supports (see Figure 9-10). You can open the Control Center directly to a selected component by selecting its entry in the K Menu Settings menu. The Settings menu displays a submenu listing the configuration categories. Select a category, and then select the component you want. For example, to configure your screen saver, select the screen saver config entry in the Desktop menu located in the Settings menu. The Control Center can be directly started by either clicking the Control Center icon in the panel or selecting Control Center from the K Menu.



Name	Description
HighColor Default	HighColor version of the default style
KDE Step	Unthemed Next-like style
KDE Step Hi-color	No description available.
KDE default	Default KDE style
KDE-SGI	Sgi-like style

□ Menubar on top of the screen in the style of MacOS
□ Apply fonts and colors to non-KDE apps

● Icons only
○ Text only
○ Text aside icons
○ Text under icons
☒ Highlight buttons under mouse
☒ Toolbars are transparent when moving

Figure 9-10: KDE Control Center

The Control Center window is divided into two panes. The left pane shows a tree view of all the components you can configure and the right pane displays the dialog windows for a selected component. On the left pane, components are arranged into categories whose titles you can expand or shrink. The Applications heading holds entries for configuring the KDE file manager's Web browser features, as well as its file management operations. Under Desktop, you can set different features for displaying and controlling your desktop. For example, the Background entry enables you to select a different background color or image for each one of your virtual desktops. Other desktop entries enable you to configure components such as the screen saver, the language used, and the window style. Some entries enable you to configure your mouse, key mappings, network connections, sound events, and window components. You can change key bindings for any of the window operations or standard operations, such as cut and paste. You can also change any of the specialized key mappings, such as the ALT-TAB key, that moves through your open applications, CTRL-TAB, which moves through your virtual desktops, or ALT-F2, which displays a dialog box for executing commands. Configuration components are actually modules. In future releases, more modules will be included as more applications and tools are added to the K Desktop. See the Help viewer for a current listing of K Desktop configuration modules.

.kde/share/config

Your **.kde** directory holds files and directories used to maintain your KDE desktop. The **.desktops** directory holds KDE desktop files whose icons are displayed on the desktop. Configuration files are located in the **.kde/share/config** directory. Here, you can find the general KDE configuration file **kfmrc**, as well as configuration files for different KDE components. **krootwmrc** holds configuration commands for your root window, **kwmrc** for the window manager, **ksoundrc** for sound, and **kcmpanelrc** for your panel. You can place configuration directives directly in any of these files-for example, to have the left mouse button display the Application Starter on the desktop, place the following lines in your **krootwmrc** file:

```
[MouseButtons]
 Left=Menu
```

MIME Types and Associated Applications

As you install new kinds of programs, they may use files of a certain type. In that case, you will need to register the type with KDE so that it can be associated with a given application or group of applications. For example, the MIME type for GIF images is **image/gif**, which is associated with image-viewing programs. You use the KDE Control Center to set up a new MIME type or to change MIME type associations with applications. Select the File Association entry under File Browsing. This will list MIME types and their associated filename extensions. Select an entry to edit it, where you can change the applications associated with it. KDE saves its MIME type information in a separate file called **mimelnk** in the KDE configuration directory.

To change the associated application for a particular file, you find the application with the file manager and then right-click it and select Edit File Type. This displays a window with two panels: General and Embedded. On the General menu, you can add an extension by clicking the Add button and entering the extension for the file type, using a * to match the prefix. For example, ***.gif** would match any file ending with **.gif**. In the Application Preference Order section, use the Add button to select the application you want associated with the file. You can have several applications for a single file, changing the preference order for them as you wish. Click the Apply button to save your changes.

KDE Directories and Files

On Red Hat, KDE is installed in the standard system directories with some variations, such as **/usr/bin** for KDE program files, **/usr/lib/kde2**, which holds KDE libraries, and **/usr/include/kde**, which contains KDE header files used in application development. (On other distributions, KDE may be installed in the **/opt/kde2** directory.)

The directories located in **/usr/share** contain files used to configure your KDE environment. The **/usr/share/mimelnk** directory maps its files to the Applications Launcher menu. Its directories and subdirectories appear as menu items and submenus on the K Menu. Their contents consist of desktop files having the extension **.desktop**, one for each menu entry. The **/usr/share/apps** file contains files and directories set up by KDE applications. **/usr/share/mimelnk** holds MIME type definitions. **/usr/share/config** contains the configuration files for particular KDE applications. For example, **kfmrc** holds configuration entries for displaying and using the Konqueror file manager. These are the system-wide defaults that can be overridden by users' own configurations in their own **.kde/share/config** directories. **/usr/share/icons** holds the default icons used on your KDE desktop and by KDE applications. The **/usr/share** directory also holds files for other configuration elements such as wallpaper, toolbars, and languages.

In the user's home directory, the **.kde** directory holds a user's own KDE configuration for the desktop and its applications. The **.kde/share** directory holds user versions of the **/usr/share** directories, specifying user configurations for menus, icons, MIME types, sounds, and applications. **.kde/share/config** holds configuration files with users' own configuration specifications for their use of KDE applications. For example, **kmailrc** holds display configurations the user set up for KMail. **.kde/share/icons** holds the icons for a user's particular themes, and **/.kde/share/sounds** holds sound files. **.kde/share/mimelnk** holds the desktop files for the menu entries in the Personal section of the Applications Launcher menu added by the user. In a user's home directory, the **.kderc** file contains current desktop configuration directives for resources such as key bindings, colors, and window styles.

Each user has a **Desktop** directory that holds KDE link files for all icons and folders on the user's desktop (see Table 9-4). These include the Trash folders and the CD-ROM and home directory links.

Table 9-4: KDE Installation Directories

System KDE Directories	**Description**
/usr/bin	KDE programs
/usr/lib/kde	KDE libraries
/usr/include/kde	Header files for use in compiling and developing KDE applications
/usr/share/config	KDE desktop and application configuration files
/usr/share/mimelnk	Desktop files used to build the K Menu
/usr/share/apps	Files used by KDE applications
/usr/share/icons	Icons used in KDE desktop and applications
/usr/share/sounds	Sounds used in KDE desktop and applications
/usr/share/doc	KDE Help system
KDEDIR	System variable that holds KDE directory (used on other distributions)
User KDE Directories	Description
.kde/AutoStart	Applications automatically started up with KDE
.kde/share/config	User KDE desktop and application configuration files for user-specified features
.kde/share/mimelnk	Desktop files used to build the user's personal menu entries on the KDE K Menu
.kde/share/apps	Directories and files used by KDE applications
.kde/share/icons	Icons used in the user's KDE desktop and applications, such as the ones for the user-specified theme
.kde/share/sounds	Sounds used in KDE desktop and applications
.kderc	User configurations directives for desktop resources
Desktop	Holds desktop files for icons and folders displayed on the user's KDE desktop
Desktop/Trash	Trash folder for files marked for deletion

System Configuration Using KDE

You can obtain full access to the system administration tools with KDE through the root desktop. Log in as the root user and start KDE. On Red Hat, the kontrol-panel window, shown in Figure 9-11, lists icons for all system configuration utilities. An icon for the Control Panel window will be displayed in the upper-left corner of the root user desktop. You can also find system tools in the System menu in the K Menu.

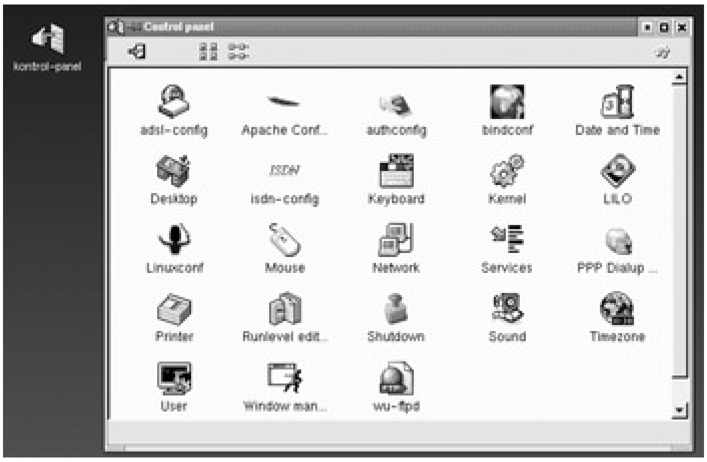


Figure 9-11: kontrol-panel window

Some of these administration tools are KDE applications, designed to work on the KDE desktop, though they will work on any X interface. For example, kuser provides a KDE interface for managing users on your system. You can add or remove users, or set permissions for current ones. The K Desktop provides two utilities for viewing and managing your processes: the KDE Task Manager (KTop) and the KDE Process Manager (kpm). The SysV Init Editor is a version of the System V Init Manager (see Chapter 23). You can use the SysV Init Editor to start and stop servers and to determine at what runlevel they will start. See the chapters on system administration, Chapters 28-34, for a more detailed discussion on administration tasks. With kppp, you can connect to the Internet using a modem. Use kppp to connect to an Internet service provider (ISP) that supports the PPP protocols. kppp is discussed in more detail in the chapter on network administration, Chapter 36. kpackage enables you to manage the RPM packages you have installed. You can use it to install new packages, see the ones that are installed, and easily display information about each package. With kpackage, you can list the files in a particular package and display the release information. You can even display text files, such as **README** and configuration files. This is an easy way to find out exactly where an application's program and configuration files are installed on your system.

Updating KDE

Currently, new versions of KDE are being released frequently, sometimes every few months. KDE releases are designed to enable users to upgrade their older versions easily. If you are a registered Red Hat customer you can use the Red Hat update utility described in Chapter 3 to update KDE automatically. Alternatively, you can download new Red Hat KDE packages from the Red Hat FTP site and install them manually as described here. The most recent (though untested) versions will be in the **rawhide** directory. Tested versions will be in the **updates** directory. Packages tailored for various distributions can be downloaded through the KDE Web site at **www.kde.org** or directly from the KDE FTP site at **ftp.kde.org** and its mirror sites in the **stable** directory. RPM packages for Red Hat distributions can be obtained from **ftp.redhat.com**.

First, log in as the root user, and then create a directory to hold the KDE files. Then connect to an FTP site. You can use a Web browser such as Netscape, but using an FTP client such as ftp, NcFTP, or even the KDE file manager is preferable. Download the files for the new version to your new directory. For Red Hat, these are a series of RPM package files. To download using ftp, be sure to turn off prompts with the **prompt** command and use **mget *** to download all the files at once. With the KDE file manager, select the files and drag them to a window open to the local directory and select Copy from the pop-up menu. Install Qt, the KDE libraries and support packages, and then the KDE base packages, followed by the remaining packages. Use the option **-Uvh** to upgrade the packages as shown here:

```
rpm -Uvh  kdebase-2.2-12.i386.rpm
```

To install a particular KDE RPM package manually, you use the **rpm** command with the **-Uvh** options. This example installs the kdenetwork package:

```
rpm -Uvh  kdenetwork-2.2.7.i386.rpm
```

Chapter 10: Window Managers

Overview

Unlike PC-based GUIs such as Windows or the Mac OS, Linux and Unix systems divide the GUI into three separate components: the X Window System, window managers, and program/file managers. The X Window System, also known as X and X11, is an underlying standardized graphic utility that provides basic graphic operations such as opening windows or displaying images. A window manager handles windowing operations such as resizing and moving windows. Window managers vary in the way windows are displayed, using different borders and window menus. All, however, use the same underlying X graphic utility. A file manager handles file operations using icons and menus, and a program manager runs programs, often allowing you to select commonly used ones from a taskbar. Unlike window managers, file and program managers can vary greatly in their capabilities. In most cases, different file and program managers can run on the same window manager.

All Linux and Unix systems use the same standard underlying X graphics utility. This means, in most cases, that an X Window System-based program can run on any of the window managers and desktops. X Window System-based software is often found at Linux or Unix FTP sites in directories labeled **X11**. You can download these packages and run them on any window manager running on your Linux system. Some may already be in the form of Linux binaries that you can download, install, and run directly. Netscape is an example. Others will be in the form of source code that can easily be configured, compiled, and installed on your system with a few simple commands. Some applications, such as Motif applications, may require special libraries.

With a window manager, you can think of a window as taking the place of a command line. Operations you perform through the window are interpreted and sent to the Linux system for execution. Window managers operate off the underlying X Window System, which actually provides the basic window operations that allow you to open, move, and close windows as well as display menus and select icons. FVWM2 and AfterStep manage these operations, each in its own way, providing their own unique interfaces. The advantage of such a design is that different window managers can operate on the same Linux system. In this sense, Linux is not tied to one type of graphical user interface (GUI). On the same Linux system, one user may be using the FVWM2 window manager, another may be using the Xview window manager, and still another the Enlightenment window manager, all at the same time. You can find out detailed information about different window managers available for Linux from the X11 Web site at **www.xwinman.org**. The site provides reviews, screenshots, and links to home sites, as well as a comparison table listing the features available for the different window managers.

Window, File, and Program Managers

With a window manager, you can use your mouse to perform windowing operations such as opening, closing, resizing, and moving windows. Several window managers are available for Linux (see Table 10-1). Some of the more popular ones are Sawfish, Enlightenment, Window Maker, AfterStep, and the Free Virtual Window Manager 2.0 (FVWM2). FVWM2 is traditionally used as the default backup window manager on most Linux systems. Window Maker and AfterStep are originally based on the NeXTSTEP interface used for the NeXT operating system. Enlightenment and Sawfish are the default window managers for Gnome by several distributions. Red Hat currently includes Enlightenment, Sawfish, and FVWM2. On Red Hat systems, Sawfish is used as the default window manager for Gnome and FVWM2 for a standard X Window System environment.

Window managers operate through the underlying X graphics utility. The X Window System actually provides the basic operations that allow you to open, move, and close windows as well as display menus and select icons. Window managers manage these operations each in their own way, providing different interfaces from which to choose. All window managers, no matter how different they may appear, use X Window System tools. In this sense, Linux is not tied to one type of graphical user interface. You can find out more about the X Window System at **www.x.org**.

Window managers originally provided only very basic window management operations such as opening, closing, and resizing of windows. Their features have been enhanced in the more sophisticated window managers such as Window Maker and Enlightenment to include support for virtual desktops, docking panels, and themes that let users change the look and feel of their desktop. However, to work with files and customize applications, you need to use file and program managers. With a file manager, you can copy, move, or erase files within different directory windows. With a program manager, you can execute commands and run programs using taskbars and program icons. A desktop program will combine the capabilities of window, file, and program managers, providing a desktop metaphor with icons and menus to run programs and access files. Gnome and the K Desktop are two such desktop programs.

Several window managers have been enhanced to include many of the features of a desktop. The window managers included with many Linux distributions have program management capabilities in addition to window handling. FVWM2 and AfterStep have a taskbar and a workplace menu that you can use to access all your X programs. With either the menu or the taskbar, you can run any X program directly from FVWM2. Window Maker provides a NeXTSTEP interface that features a docking panel for your applications with drag-and-drop support. You can drag a file to the application icon to start it with that file.

Using just a window manager, you can run any X program. Window managers have their own workspace menu and taskbar. You can also run any X program from an Xterm terminal window. There you can type the name of an X application; when you press ENTER, the X application will start up with its own window. It is best to invoke an X application as a background process by adding an ampersand (**&**) after the command. A separate window will open for the X application that you can work in.

Window Managers

Instead of the command line interface, you can use an X window manager and file manager, which will allow you to interact with your Linux system using windows, buttons, and menus. Window managers provide basic window management operations such as opening, closing, and resizing windows, and file managers allow you to manage and run programs using icons and menus. The X Window System supports a variety of window managers. See Table 10-1 for a listing of window managers and desktops along with their Web sites where you can obtain more information. You can also download current versions from these Web sites.

Windows and Icons

You run applications, display information, or list files in windows. A *window* is made up of several basic components. The *outer border* contains resize controls. Also, various buttons enable you to control the size of a window or close the window. Inside the outer border are the main components of the window: the *title bar,* which displays the name of the window; the *menu*, through which you can issue commands; and the *window pane,* which displays the contents of a window. You can change a window's size and shape using buttons and resize areas. The resize areas are the corner borders of the window. Click and hold a resize area and move the mouse to make the window larger or smaller in both height and width. You can make the window fill the whole screen using a maximize operation. Most window managers include a small button in the upper-right corner that you can click to maximize the window. To reduce the window to its original size, just click the Maximize button again. If you want to reduce the window to an icon, click the Minimize button. It's the small square with a dot in the center next to the Maximize button. Once you have reduced the window to an icon, you can reopen it later by double-clicking that icon.

Applications that have been designed as X programs will have their own menus, buttons, and even icons within their windows. You execute commands in such X applications using menus and icons. If you are running an application such as an editor, the contents of the window will be data that the menus operate on. If you are using the file manager, the contents will be icons representing files and directories. Some windows, such as terminal windows, may not have menus.

You can have several windows open at the same time. However, only one of those windows will be active. On some window managers, just moving your mouse pointer to a particular window makes it the active window, rendering all others inactive. You will need to click the title bars of others. Most window managers let the user configure the method for making the window active.

Themes

Many window managers, such as Enlightenment, Sawfish, Window Maker, AfterStep, Blackbox, and FVWM2, support themes. Themes change the look and feel for widgets on your desktop, providing different background images, animation, and sound events. With themes, users with the same window manager may have desktops that appear radically different. The underlying functionality of the window manager does not change. You can easily download themes from Web sites and install them on your window manager. New ones are constantly being added. Information and links to window manager theme sites can be found at **themes.org**.

Workspace Menu

Most window managers provide a menu through which you can start applications, perform window configurations, and exit the window manager. Window managers give this menu different names. Enlightenment calls it the applications menu, Window Maker refers to it as the root-window menu, and FVWM2 calls it the workplace menu. In this chapter, it is referred to as the *workspace menu.* This menu is usually a pop-up menu that you can display by clicking anywhere on the desktop. The mouse button you use differs with window managers. Enlightenment and Sawfish use a middle-click, whereas Window Maker uses a right-click, and FVWM2 uses a left-click. Many of the entries on this workspace menu lead to submenus that in turn may have their own submenus. For example, applications will bring up a submenu listing categories for all your X programs. Selecting the graphics item will bring up a list of all the X graphic programs on your system. If you choose Xpaint, the Xpaint program will then start up. On some window managers you will find entries for window configuration and themes. Here will be items and submenus for configuring your window manager. For example, both AfterStep and Enlightenment have menus for changing your theme (see Figures 10-3 and 10-4 for examples of workspace menus).

Desktop Areas and Virtual Desktops

Initially, you may find desktop areas disconcerting-they provide a kind of built-in enlargement feature. You will discover that the area displayed on your screen may be only part of the desktop. Moving your mouse pointer to the edge of your screen moves the screen over the hidden portions of the desktop. You will also notice a small square located on your desktop or in your window manager's icon bar, taskbar, or panel. This is called the *pager,* and you use it to view different areas of your virtual desktop. The pager will display a rectangle for every active virtual desktop. Some window managers such as FVWM2 will display only two, others such as AfterStep will have four, and others such as FVWM will start out with only one.

Each desktop rectangle will be divided into smaller squares called *desktop areas.* You can think of each desktop area as a separate extension of your desktop. It's as if you have a very large desk, only part of which is shown on the screen. The active part of the desktop is shown in the pager as a highlighted square, usually in white. This is the area of the desktop currently displayed by your screen. A desktop can have as many as 25 squares, though the default is usually 4. You can click one of the squares in the rectangle to move to that part of your desk. You could place different windows in different parts of your desk and then move to each part when you want to use its windows. In this way, everything you want on your desktop does not have to be displayed on your screen at once, cluttering it up. If you are working on the desktop and everything suddenly disappears, it may be that you accidentally clicked one of the other squares. Certain items will always be displayed on your screen, no matter what part of the virtual desktop you display. These are called "sticky" items. The pager is one, along with taskbars or panels. For example, the taskbars, icon bars, and pager will always show up on your screen no matter what part of the virtual desktop you are viewing. Windows by default are not sticky, though you can make them sticky.

Most window managers also support virtual desktops. A desktop includes all the desktop areas, along with the items displayed on them such as icons, menus, and windows. Window managers such as Enlightenment, Sawfish, AfterStep, and FVWM2 allow you to use several virtual desktops. Unlike desktop areas, which just extend a desktop, virtual desktops are separate entities. Most pagers will display the different virtual desktops as separate rectangles, each subdivided into its respective desktop areas. To move to a virtual desktop, you click its rectangle. In some window managers, such as Window Maker, you select the desktop from a list. Window managers will provide entries on their main menus for selecting virtual desktops and even moving windows from one desktop to another. Use a window manager's configuration program or configuration files to specify the number of virtual desktops you want. For example, on Sawfish you can choose the number of virtual desktops and then add as many virtual areas as you want to each, extending the area as rows and columns. Only one virtual desktop can be active at any one time.

Chapter 11: The Shell

Overview

The *shell* is a command interpreter that provides a line-oriented interactive and noninteractive interface between the user and the operating system. You enter commands on a command line, they are interpreted by the shell, and then sent as instructions to the operating system. You can also place commands in a script file to be consecutively executed much like a program (see Chapter 40). This interpretive capability of the shell provides for many sophisticated features. For example, the shell has a set of wildcard characters that can generate filenames. The shell can redirect input and output, as well as run operations in the background, freeing you to perform other tasks.

Several different types of shells have been developed for Linux: the Bourne Again shell (BASH), the Public Domain Korn shell (PDKSH), the TCSH shell, and the Z shell. All shells are available for your use, although the BASH shell is the default. You only need one type of shell to do your work. Red Hat Linux includes all the major shells, although it installs and uses the BASH shell as the default. If you use the Red Hat Linux command line shell, you will be using the BASH shell unless you specify another. This chapter discusses the BASH shell that shares many of the same features as other shells.

Note You can find out more about the BASH shell at **www.gnu.org/software/bash**. A detailed online manual is available on your Linux system using the **man** command with the **bash** keyword.

The Command Line

The Linux command line interface consists of a single line into which you enter commands with any of their options and arguments. Red Hat Linux installs with the BASH shell. From Gnome or KDE you can access the command line interface by opening a terminal window. Should you start Linux with the command line interface, you will be presented with a BASH shell command line when you log in.

By default, The BASH shell has a dollar (**$**) sign prompt, but Linux has several other types of shells, each with its own prompt. A shell *prompt*, such as the one shown here, marks the beginning of the command line:

```
$
```

The prompt designates the beginning of the command line. You are now ready to enter a command and its arguments at the prompt. In the next example, the user enters the **date** command, which displays the date. The user types the command on the first line, and then presses ENTER to execute the command.

```
$ date
Sun July 8 10:30:21 PST 2000
```

The *command line interface* is the primary interface for the shell, which interprets the commands you enter and sends them to the system. The shell follows a special syntax for interpreting the command line. The first word entered on a command line must be the name of a command. The next words are options and arguments for the command. Each word on the command line must be separated from the others by one or more spaces or tabs.

```
$ Command      Options      Arguments
```

An *option* is a one-letter code preceded by a dash that modifies the type of action the command takes. One example of a command that has options is the **ls** command. The **ls** command, with no options, displays a list of all the files in your current directory. It merely lists the name of each file with no other information.

With a **-l** option, the **ls** command modifies its task by displaying a line of information about each file, listing such data as its size and the date and time it was last modified. In the next example, the user enters the **ls** command followed by a **-l** option. The dash before the **-l** option is required. Linux uses it to distinguish an option from an argument.

```
$ ls -l
```

Another option, **-a**, lists all the files in your directory, including what are known as hidden files. *Hidden files* are often configuration files and they always have names beginning with a period. For this reason, hidden files are often referred to as *dot files*. In most cases, you can also combine options. You do so by preceding the options with an initial dash and then listing the options you want. The options **-al**, for example, list information about all the files in your directory, including any hidden files.

```
$ ls -al
```

Note Another option for the **ls** command is **-F**. With this option, the **ls** command displays directory names with a preceding slash, so you can easily identify them.

Most commands are designed to take arguments. An *argument* is a word you type in on the command line after any options. Many file management commands take filenames as their arguments. For example, if you only wanted the information displayed for a particular file, you could add that file's name after the **-l** option:

```
$ ls -l mydata
```

The shell you will start working in is the BASH shell, your default shell. This shell has special command line editing capabilities that you may find helpful as you learn Linux. You can easily modify commands you have entered before executing them, moving anywhere on the command line and inserting or deleting characters. This is particularly helpful for complex commands. You can use the CTRL-F or RIGHT ARROW keys to move forward a character, or the CTRL-B or LEFT ARROW keys to move back a character. CTRL-D or DEL deletes the character the cursor is on, and CTRL-H or BACKSPACE deletes the character before the cursor. To add text, you use the arrow keys to move the cursor to where you want to insert text and type in the new characters. At any time, you can press ENTER to execute the command. For example, if you make a spelling mistake when entering a command, rather than reentering the entire command, you can use the editing operations to correct the mistake.

Note The editing capabilities of the BASH shell command line are provided by Readline. Readline supports numerous editing operations. You can even bind a key to a selected editing operation. You can find out more about Readline in the BASH shell reference manual at **www.gnu.org/manual/bash**.

You can also use the UP ARROW key to redisplay your previously executed command. You can then reexecute that command or edit it and execute the modified command. You'll find this capability helpful when you have to repeat certain operations over and over, such as editing the same file.

Note The capability to redisplay a previous command is helpful when you've already executed a command you had entered incorrectly. In this case, you would be presented with an error message and a new, empty command line. By pressing the UP ARROW key, you can redisplay your previous command, make corrections to it, and then execute it again. This way, you would not have to enter the whole command again.

The BASH shell keeps a list, called a *history list,* of your previously entered commands. You can display each command, in turn, on your command line by pressing the UP ARROW key. The DOWN ARROW key moves you down the list. You can modify and execute any of these previous commands when you display them on your command line. This history feature is discussed in more detail in Chapter 15.

Note Some commands can be complex and take some time to execute. When you mistakenly execute the wrong command, you can interrupt and stop such commands with the interrupt keys-*CTRL-C or DEL.*

You can enter a command on several lines by typing a backslash just before you press ENTER. The backslash "escapes" the ENTER key, effectively continuing the same command line to the next line. In the next example, the **cp** command is entered on three lines. The first two lines end in a backslash, effectively making all three lines one command line.

```
$ cp -i \
mydata \
newdata
```

*Wildcards and Filename Arguments: *, ?, [ ]*

Filenames are the most common arguments used in a command. Often you may know only part of the filename, or you may want to reference several filenames that have the same extension or begin with the same characters. The shell provides a set of special characters called *wildcards* that search out, match, and generate a list of filenames. The *wildcard characters* are the asterisk, the question mark, and brackets (*****, **?**, **[]**). Given a partial filename, the shell uses these matching operators to search for files and generate a list of filenames found. The shell replaces the partial filename argument with the list of matched filenames.

This list of filenames can then become the arguments for commands such as **ls**, which can operate on many files. Table 11-1 lists the shell's wildcard characters.

Table 11-1: Shell Symbols

Common Shell Symbols	**Execution**
ENTER	Execute a command line.
;	Separate commands on the same command line.
`*command*`	Execute a command.
[]	Match on a class of possible characters in filenames.
\	Quote the following character. Used to quote special characters.
|	Pipe the standard output of one command as input for another command.
&	Execute a command in the background.
!	History command.
Wildcard Symbols	Execution
*	Match on any set of characters in filenames.
?	Match on any single character in filenames.
Redirection Symbols	Execution
>	Redirect the standard output to a file or device, creating the file if it does not exist and overwriting the file if it does exist.
>**!**	The exclamation point forces the overwriting of a file if it already exits. This overrides the **noclobber** option.
<	Redirect the standard input from a file or device to a program.
<<	Redirect the standard output to a file or device, appending the output to the end of the file.
Standard Error Redirection Symbols	Execution
2	Redirect the standard error to a file or device.
2	Redirect and append the standard error to a file or device.
2&1	Redirect the standard error to the standard output.
&	Redirect the standard error to a file or device.
|**&**	Pipe the standard error as input to another command.

The *asterisk,* *****, references files beginning or ending with a specific set of characters. You place the asterisk before or after a set of characters that form a pattern to be searched for in filenames. If the asterisk is placed before the pattern, filenames that end in that pattern are searched for. If the asterisk is placed after the pattern, filenames that begin with that pattern are searched for. Any matching filename is copied into a list of filenames generated by this operation. In the next example, all filenames beginning with the pattern "doc" are searched for and a list generated. Then all filenames ending with the pattern "day" are searched for and a list is generated. The last example shows how the * can be used in any combination of characters.

```
$ ls
doc1 doc2 document docs mydoc monday tuesday
$ ls doc*
doc1 doc2 document docs
$ ls *day
monday tuesday
$ ls m*d*
monday
$
```

Filenames often include an extension specified with a period and followed by a string denoting the file type such as **.c** for C files, **.cpp** for C++ files, or even **.jpg** for JPEG image files. The extension has no special status and is only part of the characters making up the filename. Using the asterisk makes it easy to select files with a given extension. In the next example, the asterisk is used to list only those files with a **.c** extension. The asterisk placed before the **.c** constitutes the argument for **ls**.

```
$ ls *.c
calc.c main.c
```

You can use * with the **rm** command to erase several files at once. The asterisk first selects a list of files with a given extension, or beginning or ending with a given set of characters, and then it presents this list of files to the **rm** command to be erased. In the next example, the **rm** command erases all files beginning with the pattern "doc":

```
$ rm doc*
```

The asterisk by itself matches all files. If you use a single asterisk as the argument for an **rm** command, all your files will be erased. In the next example, the **ls** * command lists all files, and the **rm** * command erases all files:

```
$ ls *
doc1 doc2 document docs mydoc myletter yourletter
$ rm *
$ ls
$
```

Use the * wildcard character carefully and sparingly with the **rm** command. The combination can be dangerous. A misplaced * in an **rm** command without the **-i** option could easily erase all your files. The first command in the next example erases only those files with a **.c** extension. The second command, however, erases all files. Notice the space between the asterisk and the period in the second command. A space in a command line functions as a *delimiter,* separating arguments. The asterisk is considered one argument, and the **.c** another argument. The asterisk by itself matches all files and, when used as an argument with the **rm** command, instructs **rm** to erase all your files.

```
$ rm *.c
$ rm * .c
```

The question mark, **?**, matches only a single incomplete character in filenames. Suppose you want to match the files **doc1** and **docA**, but not **document**. Whereas the asterisk will match filenames of any length, the question mark limits the match to just one extra character. The next example matches files that begin with the word "doc" followed by a single differing letter:

```
$ ls
doc1 docA document
$ ls doc?
doc1 docA
```

Whereas the ***** and **?** wildcard characters specify incomplete portions of a filename, the brackets, **[]**, enable you to specify a set of valid characters to search for. Any character placed within the brackets will be matched in the filename. Suppose you want to list files beginning with "doc", but only ending in *1* or *A.* You are not interested in filenames ending in *2* or *B,* or any other character. Here is how it's done:

```
$ ls
doc1 doc2 doc3 docA docB docD document
$ ls doc[1A]
doc1 docA
```

You can also specify a set of characters as a range, rather than listing them one by one. A dash placed between the upper and lower bounds of a set of characters selects all characters within that range. The range is usually determined by the character set in use. In an ASCII character set, the range "a-g" will select all lowercase alphabetic characters from *a* through *g,* inclusive. In the next example, files beginning with the pattern "doc" and ending in characters *1* through *3* are selected. Then, those ending in characters *B* through *E* are matched.

```
$ ls doc[1-3]
doc1 doc2 doc3
$ ls doc[B-E]
docB docD
```

You can combine the brackets with other wildcard characters to form flexible matching operators. Suppose you only want to list filenames ending in either a **.c** or **.o** extension, but no other extension. You can use a combination of the asterisk and brackets: *** [co]**. The asterisk matches all filenames, and the brackets match only filenames with extension **.c** or **.o**.

```
$ ls *.[co]
main.c  main.o  calc.c
```

At times, a wildcard character is actually part of a filename. In these cases, you need to quote the character by preceding it with a backslash to reference the file. In the next example, the user needs to reference a file that ends with the **?** character, **answers?**. The **?** is, however, a wildcard character and would match any filename beginning with "answers" that has one or more characters. In this case, the user quotes the **?** with a preceding backslash to reference the filename.

```
$ ls answers\?
answers?
```

Standard Input/Output and Redirection

When UNIX was designed, a decision was made to distinguish between the physical implementation and the logical organization of a file. Physically, UNIX files are accessed in randomly arranged blocks. Logically, all files are organized as a continuous stream of bytes. Linux, as a version of UNIX, has this same organization. Aside from special system calls, the user never references the physical structure of a file. To the user, all files have the same organization-a byte stream. Any file can be easily copied or appended to another because all files are organized in the same way. In this sense, only one standard type of file exists in Linux, the *byte-stream file.* Linux makes no implementational distinction between a character file and a record file, or a text file and a binary file.

This logical file organization extends to input and output operations. The data in input and output operations is organized like a file. Data input at the keyboard is placed in a data stream arranged as a continuous set of bytes. Data output from a command or program is also placed in a data stream and arranged as a continuous set of bytes. This input data stream is referred to in Linux as the *standard input*, while the output data stream is called the *standard output*. There is also a separate output data stream reserved solely for error messages, called the *standard error* (see the section on the standard error later in this chapter).

Because the standard input and standard output have the same organization as that of a file, they can easily interact with files. Linux has a redirection capability that lets you easily move data in and out of files. You can redirect the standard output so that, instead of displaying the output on a screen, you can save it in a file. You can also redirect the standard input away from the keyboard to a file, so that input is read from a file instead of from your keyboard.

When a Linux command is executed that produces output, this output is placed in the standard output data stream. The default destination for the standard output data stream is a device-in this case, the screen. *Devices*, such as the keyboard and screen, are treated as files. They receive and send out streams of bytes with the same organization as that of a byte-stream file. The screen is a device that displays a continuous stream of bytes. By default, the standard output will send its data to the screen device, which will then display the data.

For example, the **ls** command generates a list of all filenames and outputs this list to the standard output. Next, this stream of bytes in the standard output is directed to the screen device. The list of filenames is then printed on the screen. The **cat** command also sends output to the standard output. The contents of a file are copied to the standard output whose default destination is the screen. The contents of the file are then displayed on the screen.

Redirecting the Standard Output: > and >>

Suppose that instead of displaying a list of files on the screen, you would like to save this list in a file. In other words, you would like to direct the standard output to a file rather than the screen. To do this, you place the output redirection operator, > (greater-than sign), and the name of a file on the command line after the Linux command. Table 11-2 lists the different ways you can use the redirection operators. In the next example, the output of the **cat** command is redirected from the screen device to a file:

Table 11-2: The Shell Operations

Command	**Execution**
ENTER	Execute a command line.
;	Separate commands on the same command line.
command\\ *opts args*	Enter backslash before carriage return to continue entering a command on the next line.
`*command*`	Execute a command.
BACKSPACE CTRL-H	Erase the previous character.
CTRL-U	Erase the command line and start over.
CTRL-C	Interrupt and stop a command execution.
Special Characters for Filename Generation	**Execution**
*****	Match on any set of characters.
?	Match on any single characters.
[]	Match on a class of possible characters.
\\	Quote the following character. Used to quote special characters.
Redirection	Execution
command filename	Redirect the standard output to a file or device, creating the file if it does not exist and overwriting the file if it does exist.
command filename	Redirect the standard input from a file or device to a program.
command filename	Redirect the standard output to a file or device, appending the output to the end of the file.
command **!** *filename*	In the C shell and the Korn shell, the exclamation point forces the overwriting of a file if it already exits. This overrides the **noclobber** option.
command **2** *filename*	Redirect the standard error to a file or device in the Bourne shell.
command **2** *filename*	Redirect and append the standard error to a file or device in the Bourne shell.
command **2&1**	Redirect the standard error to the standard output in the Bourne shell.
command **&** *filename*	Redirect the standard error to a file or device in the C shell.
Pipes	Execution
command **|** *command*	Pipe the standard output of one command as input for another command.
command **|&** *command*	Pipe the standard error as input to another command in the C shell.
Background Jobs	Execution
&	Execute a command in the background.
fg %*jobnum*	Bring a command in the background to the foreground or resume an interrupted program.

Table 11-2: The Shell Operations

Command	Execution
bg	Place a command in the foreground into the background.
CTRL-Z	Interrupt and stop the currently running program. The program remains stopped and waiting in the background for you to resume it.
notify %*jobnum*	Notify you when a job ends.
kill %*jobnum* **kill** *proccessnum*	Cancel and end a job running in the background.
jobs	List all background jobs. The **jobs** command is not available in the Bourne shell, unless it is using the jsh shell.
ps	List all currently running processes including background jobs.
at *time date*	Execute commands at a specified time and date. The time can be entered with hours and minutes and qualified as A.M. or P.M.

```
$ cat myletter > newletter
```

The redirection operation creates the new destination file. If the file already exists, it will be overwritten with the data in the standard output. You can set the **noclobber** feature to prevent overwriting an existing file with the redirection operation. In this case, the redirection operation on an existing file will fail. You can overcome the **noclobber** feature by placing an exclamation point after the redirection operator. You can place the **noclobber** command in a shell configuration file to make it an automatic default operation (see Chapter 13). The next example sets the **noclobber** feature for the BASH shell and then forces the overwriting of the **oldletter** file if it already exists:

```
$ set -o noclobber
$ cat myletter >! oldletter
```

Although the redirection operator and the filename are placed after the command, the redirection operation is not executed after the command. In fact, it is executed before the command. The redirection operation creates the file and sets up the redirection before it receives any data from the standard output. If the file already exists, it will be destroyed and replaced by a file of the same name. In effect, the command generating the output is executed only after the redirected file has been created.

In the next example, the output of the **ls** command is redirected from the screen device to a file. First the **ls** command lists files and, in the next command, **ls** redirects its file list to the **listf** file. Then the **cat** command displays the list of files saved in **listf**. Notice the list of files in **listf** includes the **listf** filename. The list of filenames generated by the **ls** command includes the name of the file created by the redirection operation- in this case, **listf**. The **listf** file is first created by the redirection operation, and then the **ls** command lists it along with other files. This file list output by **ls** is then redirected to the **listf** file, instead of being printed on the screen.

```
$ ls
mydata intro preface
$ ls > listf
$ cat listf
mydata intro listf preface
```

Errors occur when you try to use the same filename for both an input file for the command and the redirected destination file. In this case, because the redirection operation is executed first, the input file, because it exists, is destroyed and replaced by a file of the same name. When the command is executed, it finds an input file that is empty.

In the **cat** command shown next, the file **myletter** is the name for both the destination file for redirected output and the input file for the **cat** operation. As shown in the next example, the redirection operation is executed first, destroying the **myletter** file and replacing it with a new and empty **myletter** file. Then the **cat** operation is executed and attempts to read all the data in the **myletter** file. However, nothing new is now in the **myletter** file.

```
$ cat myletter > myletter
```

You can also append the standard output to an existing file using the >> redirection operator. Instead of overwriting the file, the data in the standard output is added at the end of the file. In the next example, the **myletter** and **oldletter** files are appended to the **alletters** file. The **alletters** file will then contain the contents of both **myletter** and **oldletter**.

```
$ cat myletter >> alletters
$ cat oldletter >> alletters
```

The Standard Input

Many Linux commands can receive data from the standard input. The standard input itself receives data from a device or a file. The default device for the standard input is the keyboard. Characters typed into the keyboard are placed in the standard input, which is then directed to the Linux command. The **cat** command without a filename argument reads data from standard input. When you type in data on the keyboard, each character will be placed in the standard input and directed to the **cat** command. The **cat** command then sends the character to the standard output-the screen device-which displays the character on the screen.

If you combine the **cat** command with redirection, you have an easy way of saving what you have typed to a file. As shown in the next example, the output of the **cat** operation is redirected to the **mydat** file. The **mydat** file will now contain all the data typed in at the keyboard. The **cat** command, in this case, still has no file arguments. It will receive its data from the standard input, the keyboard device. The redirection operator redirects the output of the **cat** command to the file **mydat**. The **cat** command has no direct contact with any files; it is simply receiving input from the standard input and sending output to the standard output.

```
$ cat > mydat
This is a new line
for the cat
command
^D
$
```

Just as with the standard output, you can also redirect the standard input. The standard input may be received from a file rather than the keyboard. The operator for redirecting the standard input is the less-than sign, <. In the next example, the standard input is redirected to receive input from the **myletter** file, rather than the keyboard device. The contents of **myletter** are read into the standard input by the redirection operation. Then the **cat** command reads the standard input and displays the contents of **myletter**.

```
$ cat < myletter
hello Christopher
How are you today
$
```

You can combine the redirection operations for both standard input and standard output. In the next example, the **cat** command has no filename arguments. Without filename arguments, the **cat** command receives input from the standard input and sends output to the standard output. However, the standard input has been redirected to receive its data from a file, while the standard output has been redirected to place its data in a file.

```
$ cat < myletter > newletter
```

Pipes: |

You may find yourself in situations in which you need to send data from one command to another. In other words, you may want to send the standard output of a command to another command, not to a destination file. Suppose you want to send a list of your filenames to the printer to be printed. You need two commands to do this: the **ls** command to generate a list of filenames and the **lpr** command to send the list to the printer. In effect, you need to take the output of the **ls** command and use it as input for the **lpr** command. You can think of the data as flowing from one command to another. To form such a connection in Linux, you use what is called a *pipe*. The *pipe operator*, |, (vertical bar character) placed between two commands forms a connection between them. The standard output of one command becomes the standard input for the other. The pipe operation receives output from the command placed before the pipe and sends this data as input to the command placed after the pipe. As shown in the next example, you can connect the **ls** command and the **lpr** command with a pipe. The list of filenames output by the **ls** command is piped into the **lpr** command.

```
$ ls | lpr
```

You can combine the **pipe** operation with other shell features, such as wildcard characters, to perform specialized operations. The next example prints only files with a **.c** extension. The **ls** command is used with the asterisk and ".c" to generate a list of filenames with the **.c** extension. Then this list is piped to the **lpr** command.

```
$ ls *.c | lpr
```

In the previous example, a list of filenames was used as input, but what is important to note is pipes operate on the standard output of a command, whatever that might be. The contents of whole files or even several files can be piped from one command to another. In the next example, the **cat** command reads and outputs the contents of the **mydata** file, which are then piped to the **lpr** command:

```
$ cat mydata | lpr
```

Linux has many commands that generate modified output. For example, the **sort** command takes the contents of a file and generates a version with each line sorted in alphabetic order. The **sort** command works best with files that are lists of items. Commands such as **sort** that output a modified version of its input are referred to as *filters*. Filters are often used with pipes. In the next example, a sorted version of **mylist** is generated and piped into the **more** command for display on the screen. Note that the original file, **mylist**, has not been changed and is not itself sorted. Only the output of **sort** in the standard output is sorted.

```
$ sort mylist | more
```

You can, of course, combine several commands, connecting each pair with a pipe. The output of one command can be piped into another command, which, in turn, can pipe its output into still another command. Suppose you have a file with a list of items you want to print both numbered and in alphabetical order. To print the numbered and sorted list, you can first generate a sorted version with the **sort** command and then pipe that output to the **cat** command. The **cat** command with the **-n** option then takes as its input the sorted list and generates as its output a numbered, sorted list. The numbered, sorted list can then be piped to the **lpr** command for printing. The next example shows the command:

```
$ sort mylist | cat -n | lpr
```

The standard input piped into a command can be more carefully controlled with the standard input argument, **-**. When you use the dash as an argument for a command, it represents the standard input. Suppose you want to print a file with the name of its directory at the top. The **pwd** command outputs a directory name, and the **cat** command outputs the contents of a file. In this case, the **cat** command needs to take as its input both the file and the standard input piped in from the **pwd** command. The **cat** command will have two arguments: the standard input as represented by the dash and the filename of the file to be printed.

In the next example, the **pwd** command generates the directory name and pipes it into the **cat** command. For the **cat** command, this piped-in standard input now contains the directory name. As represented by the dash, the standard input is the first argument to the **cat** command. The **cat** command copies the directory name and the contents of the **mylist** file to the standard output, which is then piped to the **lpr** command for printing. If you want to print the directory name at the end of the file instead, simply make the dash the last argument and the filename the first argument, as in **cat mylist -**.

```
$ pwd | cat - mylist | lpr
```

Redirecting and Piping the Standard Error: >&, 2>

When you execute commands, an error could possibly occur. You may give the wrong number of arguments, or some kind of system error could take place. When an error occurs, the system issues an error message. Usually such error messages are displayed on the screen, along with the standard output. Linux distinguishes between standard output and error messages, however. Error messages are placed in yet another standard byte stream called the *standard error*. In the next example, the **cat** command is given as its argument the name of a file that does not exist, **myintro**. In this case, the **cat** command simply issues an error:

```
$ cat myintro
cat : myintro not found
$
```

Because error messages are in a separate data stream from the standard output, error messages still appear on the screen for you to see even if you have redirected the standard output to a file. In the next example, the standard output of the **cat** command is redirected to the file

mydata. However, the standard error, containing the error messages, is still directed to the screen.

```
$ cat myintro > mydata
cat : myintro not found
$
```

You can redirect the standard error as you can the standard output. This means you can save your error messages in a file for future reference. This is helpful if you need a record of the error messages. Like the standard output, the standard error has the screen device for its default destination. However, you can redirect the standard error to any file or device you choose using special redirection operators. In this case, the error messages will not be displayed on the screen.

Redirection of the standard error relies on a special feature of shell redirection. You can reference all the standard byte streams in redirection operations with numbers. The numbers 0, 1, and 2 reference the standard input, standard output, and standard error, respectively. By default, an output redirection, >, operates on the standard output, 1. You can modify the output redirection to operate on the standard error, however, by preceding the output redirection operator with the number 2. In the next example, the **cat** command again will generate an error. The error message is redirected to the standard byte stream represented by the number 2, the standard error.

```
$ cat nodata 2> myerrors
$ cat myerrors
cat : nodata not found
$
```

You can also append the standard error to a file by using the number 2 and the redirection append operator, >>. In the next example, the user appends the standard error to the **myerrors** file, which then functions as a log of errors:

```
$ cat nodata 2>> myerrors
```

Shell Variables

You define variables within a shell, and such variables are known-logically enough- as *shell variables*. Many different shells exist. Some utilities, such as the mailx utility, have their own shells with their own shell variables. You can also create your own shell using what are called *shell scripts.* You have a user shell that becomes active as soon as you log in. This is often referred to as the *login shell.* Special system variables are defined within this login shell. Shell variables can also be used to define a shell's environment, as described in Chapter 13.

Note Shell variables exist as long as your shell is active-that is, until you exit the shell. For example, logging out will exit the login shell. When you log in again, any variables you may need in your login shell must be defined once again.

Definition and Evaluation of Variables: =, $, set, unset

You define a variable in a shell when you first use the variable's name. A variable's name may be any set of alphabetic characters, including the underscore. The name may also include a number, but the number cannot be the first character in the name. A name may not have any other type of character, such as an exclamation point, an ampersand, or even a space. Such symbols are reserved by the shell for its own use. Also, a name may not include more than one word. The shell uses spaces on the command line to distinguish different components of a command such as options, arguments, and the name of the command.

You assign a value to a variable with the assignment operator, =. You type in the variable name, the assignment operator, and then the value assigned. Do not place any spaces around the assignment operator. The assignment operation **poet = Virgil**, for example, will fail. (The C shell has a slightly different type of assignment operation that is described in the section on C shell variables later in this chapter.) You can assign any set of characters to a variable. In the next example, the variable **poet** is assigned the string **Virgil**:

```
$ poet=Virgil
```

Once you have assigned a value to a variable, you can then use the variable name to reference the value. Often you use the values of variables as arguments for a command. You can reference the value of a variable using the variable name preceded by the **$** operator. The dollar sign is a special operator that uses the variable name to reference a variable's value, in effect evaluating the variable. Evaluation retrieves a variable's value, usually a set of characters. This set of characters then replaces the variable name on the command line. Wherever a **$** is placed before the variable name, the variable name is replaced with the value of the variable. In the next example, the shell variable **poet** is evaluated and its contents, **Virgil**, are then used as the argument for an **echo** command. The **echo** command simply echoes or prints a set of characters to the screen.

```
$ echo $poet
Virgil
```

You must be careful to distinguish between the evaluation of a variable and its name alone. If you leave out the **$** operator before the variable name, all you have is the variable name itself. In the next example, the **$** operator is absent from the variable name. In this case, the **echo** command has as its argument the word "poet", and so prints out "poet":

```
$ echo poet
poet
```

The contents of a variable are often used as command arguments. A common command argument is a directory path name. It can be tedious to retype a directory path that is being used over and over again. If you assign the directory path name to a variable, you can simply use the evaluated variable in its place. The directory path you assign to the variable is retrieved when the variable is evaluated with the **$** operator. The next example assigns a directory path name to a variable and then uses the evaluated variable in a copy command. The evaluation of **ldir** (which is **$ldir**) results in the path name **/home/chris/letters**. The copy command evaluates to **cp myletter /home/ chris/letters**.

```
$ ldir=/home/chris/letters
$ cp myletter $ldir
```

You can obtain a list of all the defined variables with the **set** command. The next example uses the **set** command to display a list of all defined variables and their values:

```
$ set
```

```
poet  Virgil
ldir  /home/chris/letters
$
```

If you decide you do not want a certain variable, you can remove it with the **unset** command. The **unset** command undefines a variable. The next example undefines the variable **poet**. Then the user executes the **set** command to list all defined variables. Notice that **poet** is missing.

```
$ unset poet
$ set
ldir  /home/chris/letters
$
```

Shell Scripts: User-Defined Commands

You can place shell commands within a file and then have the shell read and execute the commands in the file. In this sense, the file functions as a shell program, executing shell commands as if they were statements in a program. A file that contains shell commands is called a *shell script*.

You enter shell commands into a script file using a standard text editor such as the Vi editor. The **sh** or **.** command used with the script's filename will read the script file and execute the commands. In the next example, the text file called **lsc** contains an **ls** command that displays only files with the extension **.c**:

lsc

```
ls *.c
```

```
$ sh lsc
main.c calc.c
$ . lsc
main.c calc.c
```

You can dispense with the **sh** and **.** commands by setting the executable permission of a script file. When the script file is first created by your text editor, it is only given read and write permission. The **chmod** command with the **+x** option will give the script file executable permission. (Permissions are discussed in Chapter 13.) Once it is executable, entering the name of the script file at the shell prompt and pressing ENTER will execute the script file and the shell commands in it. In effect, the script's filename becomes a new shell command. In this way, you can use shell scripts to design and create your own Linux commands. You only need to set the permission once. In the next example, the **lsc** file's executable permission for the owner is set to *on*. Then the **lsc** shell script is directly executed like any Linux command.

```
$ chmod u+x lsc
$ lsc
main.c calc.c
```

You may have to specify that the script you are using is in your current working directory. You do this by prefixing the script name with a period and slash combination, **./**, as in **./lsc**. The period is a special character representing the name of your current working directory. The slash is a directory path name separator, as explained more fully in Chapter 12. The following example would show how you would execute the **hello** script:

```
$ ./lsc
main.c calc.c
```

Just as any Linux command can take arguments, so also can a shell script. Arguments on the command line are referenced sequentially starting with 1. An argument is referenced using the **$** operator and the number of its position. The first argument is referenced with **$1**, the second with **$2**, and so on. In the next example, the **lsext** script prints out files with a specified extension. The first argument is the extension. The script is then executed with the argument **c** (of course, the executable permission must have been set).

lsext

```
ls *.$1
```

```
$ lsext c
main.c calc.c
```

In the next example, the commands to print out a file with line numbers have been placed in an executable file called **lpnum**, which takes a filename as its argument. The **cat** command with the **-n** option first outputs the contents of the file with line numbers. Then this output is piped into the **lp** command, which prints it. The command to print out the line numbers is executed in the background.

lpnum

```
cat -n $1 | lp &
```

```
$ lpnum mydata
```

You may need to reference more than one argument at a time. The number of arguments used may vary. In **lpnum,** you may want to print out three files at one time and five files at some other time. The **$** operator with the asterisk, **$***, references all the arguments on the command line. Using **$*** enables you to create scripts that take a varying number of arguments. In the next example, **lpnum** is rewritten using **$*** so it can take a different number of arguments each time you use it:

lpnum

```
cat -n $* | lp &
```

```
$ lpnum mydata preface
```

Jobs: Background, Kills, and Interruptions

In Linux, you have control not only over a command's input and output, but also over its execution. You can run a job in the background while you execute other commands. You can also cancel commands before they have finished executing. You can even interrupt a command, starting it again later from where you left off. Background operations are particularly useful for long jobs. Instead of waiting at the terminal until a command has finished execution, you can place it in the background. You can then continue executing other Linux commands. You can, for example, edit a file while other files are printing.

Canceling a background command can often save you a lot of unnecessary expense. If, say, you execute a command to print all your files and then realize you have some large files you do not want to print, you can reference that execution of the print command and cancel it. Interrupting commands is rarely used, and sometimes it is unintentionally executed. You can, if you want, interrupt an editing session to send mail and then return to your editing session, continuing from where you left off. The background commands, as well as commands to cancel and interrupt jobs, are listed in Table 11-2.

In Linux, a command is considered a *process*-a task to be performed. A Linux system can execute several processes at the same time, just as Linux can handle several users at the same time. Commands to examine and control processes exist, though they are often reserved for system administration operations. Processes actually include not only the commands a user executes, but also all the tasks the system must perform to keep Linux running.

The commands that users execute are often called *jobs* to distinguish them from system processes. When the user executes a command, it becomes a job to be performed by the system. The shell provides a set of job control operations that enable the user to control the execution of these jobs. You can place a job in the background, cancel a job, or interrupt one.

You execute a command in the background by placing an ampersand on the command line at the end of the command. When you do so, a user job number and a system process number are displayed. The user job number, placed in brackets, is the number by which the user references the job. The system process number is the number by which the system identifies the job. In the next example, the command to print the file **mydata** is placed in the background:

```
$ lpr mydata &
[1]  534
$
```

You can place more than one command in the background. Each is classified as a job and given a name and a job number. The command **jobs** lists the jobs being run in the background. Each entry in the list consists of the job number in brackets, whether it is stopped or running, and the name of the job. The + sign indicates the job currently being processed, and the **-** sign indicates the next job to be executed. In the next example, two commands have been placed in the background. The **jobs** command then lists those jobs, showing which one is currently being executed.

```
$ lpr intro &
[1]  547
$ cat *.c > myprogs &
[2]  548
$ jobs
[1]  +  Running  lpr intro
[2]  -  Running  cat *.c > myprogs
$
```

If you wish, you can place several commands at once in the background by entering the commands on the command line, separated by an ampersand, **&**. In this case, the **&** both separates commands on the command line and executes them in the background. In the next example, the first command, to sort and redirect all files with a **.l** extension, is placed in the background. On the same command line, the second command, to print all files with a **.c** extension, is also placed in the background. Notice the two commands each end with **&**. The **jobs** command then lists the **sort** and **lpr** commands as separate operations.

```
$ sort *.l > ldocs &; lpr *.c &
[1]  534
[2]  567
$ jobs
[1]  +  Running  sort *.l > ldocs
[2]  -  Running  lpr
$
```

After you execute any command in Linux, the system tells you what background jobs, if you have any running, have been completed so far. The system does not interrupt any operation, such as editing, to notify you about a completed job. If you want to be notified immediately when a certain job ends, no matter what you are doing on the system, you can use the **notify** command to instruct the system to tell you. The **notify** command takes a job number as its argument. When that job is finished, the system interrupts what you are doing to notify you the job has ended. The next example tells the system to notify the user when job 2 has finished:

```
$ notify %2
```

You can bring a job out of the background with the foreground command, **fg**. If only one job is in the background, the **fg** command alone will bring it to the foreground. If more than one job is in the background, you must use the job's number with the command. You place the job number after the **fg** command, preceded with a percent sign. A **bg** command also places a job in the background. This command is usually used for interrupted jobs. In the next example, the second job is brought back into the foreground. You may not immediately receive a prompt again because the second command is now in the foreground and executing. When the command is finished executing, the prompt appears and you can execute another command.

```
$ fg %2
cat *.c > myprogs
$
```

If you want to stop a job running in the background, you can force it to end with the **kill** command. The **kill** command takes as its argument either the user job number or the system process number. The user job number must be preceded by a percent sign, **%**. You can find out the job number from the **jobs** command. In the next example, the **jobs** command lists the background jobs; then job 2 is canceled:

```
$ jobs
[1]  +  Running  lpr intro
[2]  -  Running  cat *.c > myprogs
$ kill %2
$
```

You can also cancel a job using the system process number, which you can obtain with the **ps** command. The **ps** command displays a great deal more information than the **jobs** command does. It is discussed in detail in Chapter 29 on systems administration. The next example lists the processes a user is running. The PID is the system process number, also known as the process ID. TTY is the terminal identifier. The time is how long the process has taken so far. COMMAND is the name of the process.

```
$ ps
PID     TTY       TIME      COMMAND
523     tty24     0:05      sh
567     tty24     0:01      lpr
570     tty24     0:00      ps
```

You can then reference the system process number in a **kill** command. Use the process number without any preceding percent sign. The next example kills process 567:

```
$ kill 567
```

You can suspend a job and stop it with the CTRL-Z keys. This places the job to the side until it is restarted. The job is not ended; it merely remains suspended until you want to continue. When you're ready, you can continue with the job in either the foreground or the background using the **fg** or **bg** command. The **fg** command restarts a suspended job in the foreground. The **bg** command places the suspended job in the background.

At times, you may need to place a currently running job in the foreground into the background. However, you cannot move a currently running job directly into the background. You first need to suspend it with CTRL-Z, and then place it in the background with the **bg** command. In the next example, the current command to list and redirect **.c** files is first suspended with a CTRL-Z. Then that job is placed in the background.

```
$ cat *.c > myprogs
^Z
$ bg
```

Filters and Regular Expressions

Filters are commands that read data, perform operations on that data, and then send the results to the standard output. Filters generate different kinds of output, depending on their task. Some filters only generate information about the input, other filters output selected parts of the input, and still other filters output an entire version of the input, but in a modified way. Some filters are limited to one of these, while others have options that specify one or the other. You can think of a filter as operating on a stream of data- receiving data and generating modified output. As data is passed through the filter, it is analyzed, screened, or modified.

The data stream input to a filter consists of a sequence of bytes that can be received from files, devices, or the output of other commands or filters. The filter operates on the data stream, but it does not modify the source of the data. If a filter receives input from a file, the file itself is not modified. Only its data is read and fed into the filter.

The output of a filter is usually sent to the standard output. It can then be redirected to another file or device, or piped as input to another utility or filter. All the features of redirection and pipes apply to filters. Often data is read by one filter and its modified output piped into another filter.

Note Data could easily undergo several modifications as it is passed from one filter to another. However, it is always important to realize the original source of the data is never changed.

Many utilities and filters use patterns to locate and select specific text in your file. Sometimes, you may need to use patterns in a more flexible and powerful way, searching for several different variations on a given pattern. You can include a set of special characters in your pattern to enable a flexible search. A pattern that contains such special characters is called a *regular expression*. Regular expressions can be used in most filters and utilities that employ pattern searches such as Ed, **sed**, **awk**, **grep**, and **egrep**.

Tip Although many of the special characters used for regular expressions are similar to the shell wildcard characters, they are used in a different way. Shell wildcard characters operate on filenames. Regular expressions search text.

In Linux, as in UNIX, text files are organized into a series of lines. For this reason, many editors and filters are designed to operate on a text file line by line. The first UNIX editor, *Ed,* is a line editor whose commands reference and operate on a text file one line at a time. Other editing utilities and filters operate on text much the same way as the Ed line editor. In fact, the Ed editor and other editing filters use the same set of core line editing commands. The editing filters such as **sed** and **diff** use those same line editing commands to edit filter input. An edit filter receives lines of text as its input and performs line editing operations on them, outputting a modified version of the text. Three major edit filters exist: **tr**, which translates characters; **diff**, which outputs editing information about two files; and **sed**, which performs line editing operations on the input. Table 11-4 lists the different editing filters.

Using Redirection and Pipes with Filters

Filters send their output to the standard output and so, by default, display their output on the screen. The simplest filters merely output the contents of files. You have already seen the **cat** commands. What you may not have realized is that **cat** is a filter. It receives lines of data and outputs a version of that data. The **cat** filter receives input and copies it out to the standard output, which, by default, is displayed on the screen. Commonly used filters are listed in Tables 11-3 and 11-4.

Table 11-3: Filters

Command	**Execution**
cat *filenames*	Displays a file. It can take filenames for its arguments. It outputs the contents of those files directly to the standard output, which, by default, is the screen.
tee *filename*	Copies the standard input to a file while sending it on to the standard output. It is usually used with another filter and enables you to save output to a file while sending the output on to another filter or utility.
head *filename*	Displays the first few lines of a file. The default is ten lines,

Table 11-3: Filters

Command	**Execution**
	but you can specify the number of lines.
tail *filename*	Displays the last lines in a file. The default is ten lines, but you can specify the number of lines **$ tail** *filenames.*
wc *filename*	Counts the number of lines, words, and characters in a file and outputs only that number.
c	Counts the number of characters in a file.
l	Counts the number of lines in a file.
w	Counts the number of words in a file.
spell *filename*	Checks the spelling of each word in a file and outputs only the misspelled words.
sort *filename*	Outputs a sorted version of a file.
cmp *filename filename*	Compares two files, character by character, checking for differences. Stops at the first difference it finds and outputs the character position and line number.
comm. *filename filename*	Compares two files, line by line, and outputs both files according to lines that are similar and different for each.
grep *pattern filenames*	Searches files for a pattern and lists any matched lines.
i	Ignores uppercase and lowercase differences.
c	Only outputs a number-the count of the lines with the pattern.
l	Displays the names of the files that contain the matching pattern.
n	Outputs the line number along with the text of those lines with the matching pattern.
v	Outputs all those lines that do not contain the matching pattern.
fgrep *patterns file-list*	Searches files in the file list for several patterns at the same time. Executes much faster than either **grep** or **egrep;** however, **fgrep** cannot interpret special characters and cannot search for regular expressions.
egrep *pattern file-list*	Searches files in the file list for the occurrence of a pattern. Like **fgrep**, it can read patterns from a file. Like **grep**, it can use regular expressions, interpreting special characters. However, unlike **grep**, it can also interpret extended special characters such as **?**, **|**, and **+**.
pr	Outputs a paginated version of the input, adding headers, page numbers, and any other specified format.
cpio *generated-filenames* **|** **cpio -o** *archive-file* **cpio -i** *filenames archive-file*	Copies files to an archive and extracts files from an archive. Has two modes of operation: one using the **-o** option to copy files to an archive and the other using the **-i** option to extract files from an archive. When copying files to an archive, you need first to generate the list of filenames using a command such as **ls** or **find**.

Table 11-4: Edit Filters

Command	**Execution**
sed *editing-command file-list*	Outputs an edited form of its input. **sed** takes as an argument an editing command and a file list. The editing command is executed on input read from files in the file list. **sed** then outputs an edited version of the files. The editing commands are line editing commands similar to those used for the Ed line editor.
n	With this option, **sed** does not output lines automatically. This option is usually used with the **p** command to output only selected lines.
f *filename*	With this option, **sed** reads editing commands *filename.*
Line Commands	**(You need to quote any newline characters if you are entering more than one line)**
a	Appends text after a line.
i	Inserts text before a line.
c	Changes text.
d	Deletes lines.
p	Prints lines.
w	Writes lines to a file.
r	Reads lines from a file.
q	Quits the **sed** editor before all lines are processed.
n	Skips processing to next line.
s/*pattern*/*replacement*/	Substitutes matched pattern with replacement text.
g s/*pat*/*rep*/**g**	Global substitution on a line.
p s/*pat*/*rep*/**p**	Outputs the modified line.
w s/*pat*/*rep*/**w** *fname*	Writes the modified line to a file.
/*pattern*/	A line can be located and referenced by a pattern.
diff *filename filename*	Compares two files and outputs the lines that are different as well as the editing changes needed to make the first file the same as the second file.
f1-linenum **a** *f2-line1, f2-line2*	Appends lines from file2 to after *f1-linenum* in file1.
f1-line1, f1-line2 **d** *f1-linenum*	Deletes the lines in file1.
f1-line1, f1-line2 **c** *f2-line1, f2-line2*	Replaces lines in file1 with lines in file2.
b	Ignores any trailing or duplicate blank.
c	Outputs a context for differing lines. Three lines above and below are displayed.
e	Outputs a list of Ed editing commands that, when executed, change the first file into an exact copy of the second file.
tr *first-character-list*	Outputs a version of the input in which characters in the first

Table 11-4: Edit Filters

Command	Execution
second-character-list	character list that occur in the input are replaced in the output by corresponding characters in the second character list.

You can save the output of a filter in a file or send it to a printer. To do so, you need to use redirection or pipes. To save the output of a filter to a file, you redirect it to a file using the redirection operation, >. To send output to the printer, you pipe the output to the lpr utility, which then prints it. In the next command, the **cat** command pipes its output to the **lpr** command, which then prints it.

```
$ cat complist | lpr
```

Other commands for displaying files, such as **more**, may seem to operate like a filter, but they are not filters. You need to distinguish between filters and device-oriented utilities, such as **lpr** and **more**. Filters send their output to the standard output. A device-oriented utility such as **lpr**, though it receives input from the standard input, sends its output to a device. In the case of **lpr**, the device is a printer; for **more**, the device is the terminal. Such device-oriented utilities may receive their input from a filter, but they can only output to their device.

All filters accept input from the standard input. In fact, the output of one filter can be piped as the input for another filter. Many filters also accept input directly from files, however. Such filters can take filenames as their arguments and read data directly from those files. The **cat** and **sort** filters operate in this way. They can receive input from the standard input or use filename arguments to read data directly from files.

One of the more powerful features of **cat** is it can combine the contents of several files into one output stream. This output can then be piped into a utility or even another filter, allowing the utility or filter to operate on the combined contents of files as one data stream. For example, if you want to view the contents of several files at once, screen by screen, you must first combine them with the **cat** filter and then pipe the combined data into the **more** filter. The **more** command is, then, receiving its input from the standard input. In the following set of examples, the **cat** filter copies the contents of **preface** and **intro** into a combined output. In the first example, this output is piped into the **more** command. The **more** filter then enables you to view the combined text, screen by screen. In the second, the output is piped to the printer using the **lpr** command and, in the third, the output is redirected to a file called **frontdata**.

```
$ cat preface intro | more
$ cat preface intro | lpr
$ cat preface intro > frontdata
```

Types of Filter Output: wc, spell, and sort

The output of a filter may be a modified copy of the input, selected parts of the input, or simply some information about the input. Some filters are limited to one of these, while others have options that specify one or the other. The **wc**, **spell**, and **sort** filters illustrate all three kinds of output. The **wc** filter merely prints counts of the number of lines, words, and characters in a file. The **spell** filter selects misspelled words and outputs only those words.

The **sort** filter outputs a complete version of the input, but in sorted order. These three filters are listed in Table 11-3, along with their more commonly used options.

The **wc** filter takes as its input a data stream, which is usually data read from a file. It then counts the number of lines, words, and characters (including the newline character, found at the end of a line) in the file and simply outputs these counts. In the next example, the **wc** command is used to find the number of lines, words, and characters in the **preface** file:

```
$ wc preface
6      27     142   preface
```

The **spell** filter checks the spelling of words in its input and outputs only the misspelled.

```
$ spell foodlistsp
soop
vegetebels
```

Using redirection, you can save those words in a file. With a pipe, you can print them. In the next example, the user saves the misspelled words to a file called **misspell**:

```
$ spell foodlistsp > misspell
```

You can pipe the output of one filter into another filter, in effect, applying the capabilities of several filters to your data. For example, suppose you only want to know how many words are misspelled. You could pipe the output of the **spell** filter into the **wc** filter, which would count the number of misspelled words. In the next example, the words in the **foodlistsp** file are spell-checked, and the list of misspelled words is piped to the **wc** filter. The **wc** filter, with its **-w** option, then counts those words and outputs the count.

```
$ spell preface | wc -w
2
```

The **sort** filter outputs a sorted version of a file. **sort** is a useful utility with many different sorting options. These options are primarily designed to operate on files arranged in a database format. In fact, **sort** can be thought of as a powerful data manipulation tool, arranging records in a database-like file. This chapter examines how **sort** can be used to alphabetize a simple list of words. The **sort** filter sorts, character by character, on a line. If the first character in two lines is the same, **sort** will sort on the next character in each line. You can, of course, save the sorted version in a file or send it to the printer. In the next example, the user saves the sorted output in a file called **slist**:

```
$ sort foodlist > slist
```

Searching Files: grep and fgrep

The **grep** and **fgrep** filters search the contents of files for a pattern. They then inform you what file the pattern was found in and print the lines in which it occurred in each file. Preceding each line is the name of the file in which the line is located. **grep** can search for only one pattern, whereas **fgrep** can search for more than one pattern at a time. The **grep** and **fgrep** filters, along with their options, are described in Table 11-3.

The **grep** filter takes two types of arguments. The first argument is the pattern to be searched for; the second argument is a list of filenames, which are the files to be searched. You enter the filenames on the command line after the pattern. You can also use special characters, such as the asterisk, to generate a file list.

```
$ grep pattern filenames-list
```

In the next example, the **grep** command searches the lines in the **preface** file for the pattern "stream":

```
$ cat preface
A text file in Unix
consists of a stream of
characters.  An editor can
be used to create such
text files, changing or
adding to the character
data in the file.
$ grep stream preface
 consists of a stream of
```

If you want to include more than one word in the pattern search, you enclose the words within single quotation marks. This is to quote the spaces between the words in the pattern. Otherwise, the shell would interpret the space as a delimiter or argument on the command line, and **grep** would try to interpret words in the pattern as part of the file list. In the next example, **grep** searches for the pattern "text file":

```
$ grep 'text file' preface
A text file in Unix
text files, changing or
```

If you use more than one file in the file list, **grep** will output the name of the file before the matching line. In the next example, two files, **preface** and **intro**, are searched for the pattern "data". Before each occurrence, the filename is output.

```
$ grep data preface intro
 preface: data in the file.
 intro: new data
```

As mentioned earlier, you can also use shell wildcard characters to generate a list of files to be searched. In the next example, the asterisk wildcard character is used to generate a list of all files in your directory. This is a simple way of searching all of a directory's files for a pattern.

```
$ grep data *
```

The special characters are often useful for searching a selected set of files. For example, if you want to search all your C program source code files for a particular pattern, you can specify the set of source code files with ***.c**. Suppose you have an unintended infinite loop in your program and you need to locate all instances of iterations. The next example searches only those files with a **.c** extension for the pattern "while" and displays the lines of code that perform iterations:

```
$ grep while *.c
```

Regular Expressions

Regular expressions enable you to match possible variations on a pattern, as well as patterns located at different points in the text. You can search for patterns in your text that have different ending or beginning letters, or you can match text at the beginning or end of a line. The regular expression special characters are the circumflex, dollar sign, asterisk, period, and brackets: ^, **$**, *, **.**, **[]**. The circumflex and dollar sign match on the beginning and end of a line. The asterisk matches repeated characters, the period matches single characters, and the brackets match on classes of characters. See Table 11-5 for a listing of the regular expression special characters.

Table 11-5: Regular Expression Special Characters

Character	**Match**	**Operation**
^	Start of a line	References the beginning of a line
$	End of a line	References the end of a line
.	Any character	Matches on any one possible character in a pattern
*	Repeated characters	Matches on repeated characters in a pattern
[]	Classes	Matches on classes of characters (a set of characters) in the pattern

Note Regular expressions are used extensively in many Linux filters and applications to perform searches and matching operations. The Vi and Emacs editors and the **sed**, **diff**, **grep,** and **gawk** filters all use regular expressions.

To match on patterns at the beginning of a line, you enter the ^ followed immediately by a pattern. The ^ special character makes the beginning of the line an actual part of the pattern to be searched. In the next example, **^consists** matches on the line beginning with the pattern "consists":

```
^consists
consists of a stream of
```

The next example uses the **$** special character to match patterns at the end of a line:

```
such$
 be used to create such
```

The *period* is a special character that matches any one character. Any character will match a period in your pattern. The pattern **b.d** will find a pattern consisting of three letters. The first letter will be *b*, the third letter will be *d*, and the second letter can be any character. It will match on "bid", "bad", "bed", "b+d", or "b d", for example. Notice the space is a valid character (so is a tab).

For the period special character to have much effect, you should provide it with a context-a beginning and ending pattern. The pattern **b.d** provides a context consisting of the preceding *b* and the following *d*. If you specified **b.** without a *d*, then any pattern beginning with *b* and having at least one more character would match. The pattern would match on "bid", "bath", "bedroom", and "bump", as well as "submit", "habit", and "harbor".

The asterisk special character, *, matches on zero or more consecutive instances of a character. The character matched is the one placed before the asterisk in the pattern. You can think of the asterisk as an operator that takes the preceding character as its operand. The asterisk will search for any repeated instances of this character. Here is the syntax of the asterisk special character:

```
c*    matches on zero or more repeated occurrences of whatever
      the character c is:
c cc ccc cccc  and so on.
```

The asterisk comes in handy when you need to replace several consecutive instances of the same character. The next example matches on a pattern beginning with *b* and followed by consecutive instances of the character *o*. This regular expression will match on "boooo", "bo", "boo", and "b".

```
bo*
    book
    born
    booom
    zoom    no match
```

The **.*** pattern used by itself will match on any character in the line; in fact, it selects the entire line. If you have a context for **.***, you can match different segments of the line. A pattern placed before the **.*** special characters will match the remainder of the line from the occurrence of the pattern. A pattern placed after the **.*** will match the beginning of the line up until the pattern. The **.*** placed between patterns will match any intervening text between those patterns on the line. In the next example, the pattern **.*and** matches everything in the line from the beginning up to and including the letters "and". Then the pattern **and.*** matches everything in the line from and including the letters "and" to the end of the line. Finally, the pattern /**o.*F**/ matches all the text between and including the letters *o* and *F*.

```
.*and    Hello to you and to them Farewell

and.*    Hello to you and to them Farewell

o.*F    Hello to you and to them Farewell
```

Note Because the * special character matches zero or more instances of the character, you can provide a context with zero intervening characters. For example, the pattern **I.*t** matches on "It" as well as "Intelligent".

Suppose instead of matching on a specific character or allowing a match on any character, you need to match only on a selected set of characters. For example, you might want to match on words ending with an *A* or *H*, as in "seriesA" and "seriesH", but not "seriesB" or "seriesK". If you used a period, you would match on all instances. Instead, you need to specify that *A* and *H* are the only possible matches. You can do so with the brackets special characters.

You use the brackets special characters to match on a set of possible characters. The characters in the set are placed within brackets and listed next to each other. Their order of listing does not matter. You can think of this set of possible characters as defining a class of characters, and characters that fall into this class are matched. You may notice the brackets operate much like the shell brackets. In the next example, the user searches for a pattern beginning with "doc" and ending with either the letters *a*, *g*, or *N*. It will match on "doca", "docg", or "docN", but not on "docP".

```
doc[agN]
    List of documents
    doca docb
    docg docN docP
```

The brackets special characters are particularly useful for matching on various suffixes or prefixes for a pattern. For example, suppose you need to match on filenames that begin with the pattern "week" and have several different suffixes, as in **week1**, **week2**, and so on. To match on just those files with suffixes 2, 4, and 5, you enclose those characters within brackets. In the next example, notice the pattern **week[245]** matches on **week2** and **week4**, but not on **week1**:

```
week[245]
    week2 weather
    reports on week4
    week1 reports        no match
```

The brackets special characters are also useful for matching on a pattern that begins in either uppercase or lowercase. Linux distinguishes between uppercase and lowercase characters. The pattern "computer" is different from the pattern "Computer"; "computer" would not match on the version beginning with an uppercase *C*. To match on both patterns, you need to use the brackets special characters to specify both *c* and *C* as possible first characters in the pattern. Place the uppercase and lowercase versions of the same character within brackets at the beginning of the pattern. For example, the pattern **[Cc]omputer** searches for the pattern "computer" beginning with either an uppercase *C* or a lowercase *c*.

You can specify a range of characters within the brackets with the dash. Characters are ranged according to the character set being used. In the ASCII character set, lowercase letters are grouped together. Specifying a range with **[a-z]** selects all the lowercase letters. In the first example, shown next, any lowercase letter will match the pattern. More than one range can be specified by separating the ranges with a comma. The ranges **[A-Za-z]** select all alphabetic letters, both uppercase and lowercase.

```
doc[a-z]        doca docg docN docP
doc[A-Za-z]     doca docg docN docP
```

Although shell file matching characters enable you to match on filenames, regular expressions enable you to match on data within files. Using **grep** with regular expressions, you can locate files and the lines in them that match a specified pattern. You can use special characters in a **grep** pattern, making the pattern a regular expression. **grep** regular expressions use the *****, **.**, and **[]** special characters, as well as the ^ and **$** special characters.

Suppose you want to use the long-form output of **ls** to display just your directories. One way to do this is to generate a list of all directories in the long form and pipe this list to **grep**, which can then pick out the directory entries. You can do this by using the ^ special character to specify the beginning of a line. Remember, in the long-form output of **ls**, the first character indicates the file type. A **d** represents a directory, an **l** represents a symbolic link, and an **a** represents a regular file. Using the pattern **'^d'**, **grep** will match only on those lines beginning with a *d*.

```
$ ls -l | grep '^d'
drwxr-x---   2  chris 512 Feb 10 04:30  reports
drwxr-x---   2  chris 512 Jan 6  01:20  letters
```

If you only want to list those files that have symbolic links, you can use the pattern **^l**:

```
$ ls -l | grep '^l'
lrw-rw-r-- 1  chris  group 4    Feb 14   10:30  lunch
```

Be sure to distinguish between the shell wildcard character and special characters used in the pattern. When you include special characters in your **grep** pattern, you need to quote the pattern. Notice regular-expression special characters and shell wildcard characters use the same symbols: the asterisk, period, and brackets. If you do not, then any special characters in the pattern will be interpreted by the shell as shell wildcard characters. Without quotes, an asterisk would be used to generate filenames rather than being evaluated by **grep** to search for repeated characters. Quoting the pattern guarantees that **grep** will evaluate the special characters as part of a regular expression. In the next example, the asterisk special character is used in the pattern as a regular expression and in the filename list as a shell wildcard character to generate filenames. In this case, all files in the current directory will be searched for patterns with zero or more *s*'s after "report":

```
$ grep 'reports*' *
mydata: The report was sitting on his desk.
weather: The weather reports were totally accurate.
```

The brackets match on either a set of characters, a range of characters, or a nonmatch of those characters. For example, the pattern **doc[abc]** matches on the patterns "doca", "docb", and "docc", but not on "docd". The same pattern can be specified with a range: **doc[a-c]**. However, the pattern **doc[^ab]** will match on any pattern beginning with "doc" but not ending in *a* or *b*. Thus, "docc" will be retrieved, but not "doca" or "docb". In the next example, the user finds all lines that reference "doca", "docb", or **"**docc":

```
$ grep 'doc[abc]' myletter
File letter doca and docb.
We need to redo docc.
```

Certain Linux utilities, such as **egrep** and **awk**, can make use of an extended set of special characters in their patterns. These special characters are |, **( )**, +, and **?**, and are listed in Table 11-6.

Table 11-6: Full Regular Expression Special Characters

Character	**Execution**
pattern|*pattern*	Logical OR for searching for alternative patterns
(*pattern*)	Parentheses for grouping patterns
char+	Searches for one or more repetitions of the previous character
char?	Searches for zero or one instance of the previous character

The + and **?** are variations on the * special character, whereas | and **( )** provide new capabilities. Patterns that can use such special characters are referred to as full regular expressions. The Ed and Ex standard line editors do not have these extended special characters. Only **egrep**, which is discussed here, and **awk** have extended special characters.

The + sign matches one or more instances of a character. For example, **t+** matches at least one or more *t*'s, just as **tt*** does. **t+** matches on "sitting" or "biting", but not "ziing". The **?** matches zero or one instance of a character. For example, **t?** matches on one *t* or no *t*'s, but not "tt". The expression **it?i** matches on "ziing" and "biting", but not "sitting". With the + special character, the regular expression **an+e** matches on one or more instances of *n* preceded by *a* and followed by *e*. The "ane" is matched in "anew", and "anne" is matched on "canned".

The **|** and **( )** special characters operate on pattern segments, rather than just characters. The **|** is a logical OR special character that specifies alternative search patterns within a single regular expression. Although part of the same regular expression, the patterns are searched for as separate patterns. The search pattern **create|stream** searches for either the pattern "create" or "stream".

```
create|stream
 consists of a stream of
 be used to create such
```

The **egrep** command combines the capabilities of **grep** and **fgrep**. Like **fgrep**, it can search for several patterns at the same time. Like **grep**, it can evaluate special characters in its patterns and search for regular expressions. Unlike **grep**, however, it can evaluate extended special characters, such as the logical OR operator, **|**. In this respect, **egrep** is the most powerful of the three search filters.

To search for several patterns at once, you can either enter them on the command line separated by a newline character as **fgrep** does, or you can use the logical OR special character in a pattern to specify alternative patterns to be searched for in a file. The patterns are actually part of the same regular expression, but they are searched for as separate patterns. The pattern **create|stream egrep** will search for either the pattern "create" or the pattern "stream".

```
$ egrep 'create|stream' preface
consists of a stream of
 be used to create such
```

Chapter 12: The Linux File Structure

Overview

In Linux, all files are organized into directories that, in turn, are hierarchically connected to each other in one overall file structure. A file is referenced not just according to its name, but also according to its place in this file structure. You can create as many new directories as you want, adding more directories to the file structure. The Linux file commands can perform sophisticated operations, such as moving or copying whole directories along with their subdirectories. You can use file operations such as **find**, **cp**, **mv**, and **ln** to locate files and copy, move, or link them from one directory to another. Desktop file managers, such as konqueror, Nautilus, and Midnight Commander used on the KDE and Gnome desktops, provide a graphical user interface to perform the same operations using icons, windows, and menus (See Chapters 9 and 10). This chapter will focus on the commands you use in the shell command line to manage files, such as **cp** and **mv**. However, whether you use the command line or a GUI file manager, the underlying file structure is the same.

Together, these features make up the Linux file structure. This chapter first examines different types of files, as well as file classes. Then, the chapter examines the overall Linux file structure and how directories and files can be referenced using pathnames and the working directory. The last part of the chapter discusses the different file operations such as copying, moving, and linking files, as well as file permissions. The organization of the Linux file structure into its various system and network administration directories is discussed in detail in Chapter 32.

Linux Files

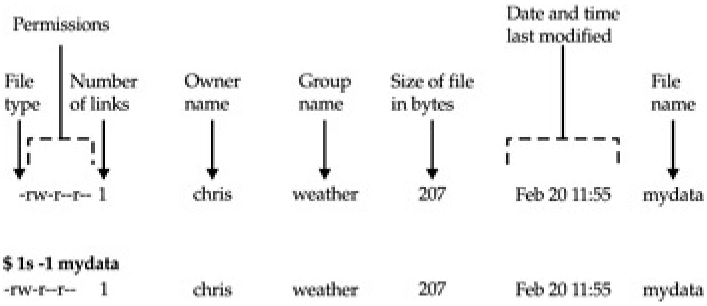
You can name a file using any alphabetic characters, underscores, and numbers. You can also include periods and commas. Except in certain special cases, you should never begin a filename with a period. Other characters, such as slashes, question marks, or asterisks, are reserved for use as special characters by the system and should not be part of a filename. Filenames can be as long as 256 characters.

You can include an extension as part of a filename. A period is used to distinguish the filename proper from the extension. Extensions can be useful for categorizing your files. You are probably familiar with certain standard extensions that have been adopted by convention. For example, C source code files always have an extension of **.c**. Files that contain compiled object code have a **.o** extension. You can, of course, make up your own file extensions. The following examples are all valid Linux filenames:

```
preface
chapter2
9700info
New_Revisions
calc.c
intro.bk1
```

Special initialization files are also used to hold shell configuration commands. These are the hidden, or dot, files referred to in Chapter 5 that begin with a period. Dot files used by commands and applications have predetermined names. Recall that when you use **ls** to display your filenames, the dot files will not be displayed. To include the dot files, you need to use **ls** with the **-a** option. Dot files are discussed in more detail in the chapter on shell configuration, Chapter 13.

As shown in Figure 12-1, the **ls -l** command displays detailed information about a file. First the permissions are displayed, followed by the number of links, the owner of the file, the name of the group the user belongs to, the file size in bytes, the date and time the file was last modified, and the name of the file. Permissions indicate who can access the file: the user, members of a group, or all other users. Permissions are discussed in detail later in this chapter. The group name indicates the group permitted to access the file object. In Figure 12-1, the file type for **mydata** is that of an ordinary file. Only one link exists, indicating the file has no other names and no other links. The owner's name is **chris**, the same as the login name, and the group name is **weather**. Other users probably also belong to the **weather** group. The size of the file is 207 bytes, and it was last modified on February 20 at 11:55 A.M. The name of the file is **mydata**.



```
$ ls -l mydata
-rw-r--r--   1        chris     weather     207      Feb 20 11:55   mydata
```

Figure 12-1: File information displayed using the **-l** option for the **ls** command

If you want to display this detailed information for all the files in a directory, simply use the **ls -**l command without an argument.

```
$ ls -l
-rw-r--r-- 1 chris weather 207 Feb 20 11:55 mydata
-rw-rw-r-- 1 chris weather 568 Feb 14 10:30 today
-rw-rw-r-- 1 chris weather 308 Feb 17 12:40 monday
```

All files in Linux have one physical format-a byte stream. A *byte stream* is just a sequence of bytes. This allows Linux to apply the file concept to every data component in the system. Directories are classified as files, as are devices. Treating everything as a file allows Linux to organize and exchange data more easily. The data in a file can be sent directly to a device such as a screen because a device interfaces with the system using the same byte-stream file format as regular files.

This same file format is used to implement other operating system components. The interface to a device, such as the screen or keyboard, is designated as a file. Other components, such as directories, are themselves byte-stream files, but they have a special internal organization. A directory file contains information about a directory, organized in a special directory format. Because these different components are treated as files, they can be said to constitute different *file types.* A character device is one file type. A directory is another file type. The number of these file types may vary according to your specific implementation of Linux. Five common types of files exist, however: ordinary files, directory files, first-in first-out pipes, character device files, and block device files. Although you may rarely reference a file's type, it can be useful when searching for directories or devices. Later in the chapter, you see how to use the file type in a search criterion with the **find** command to search specifically for directory or device names.

Although all ordinary files have a byte-stream format, they may be used in different ways. The most significant difference is between binary and text files. Compiled programs are examples of binary files. However, even text files can be classified according to their different uses. You can have files that contain C programming source code or shell commands, or even a file that is empty. The file could be an executable program or a directory file. The Linux **file** command helps you determine for what a file is used. It examines the first few lines of a file and tries to determine a classification for it. The **file** command looks for special keywords or special numbers in those first few lines, but it is not always accurate. In the next example, the **file** command examines the contents of two files and determines a classification for them:

```
$ file monday reports
monday: text-
reports: directory
```

If you need to examine the entire file byte by byte, you can do so with the **od** (octal dump) command. The **od** command performs a dump of a file. By default, it prints every byte in its octal representation. However, you can also specify a character, decimal, or hexadecimal representation. The **od** command is helpful when you need to detect any special character in your file, or if you want to display a binary file. If you perform a character dump, then certain nonprinting characters will be represented in a character notation. For example, the carriage return is represented by a **\n**. Both the **file** and **od** commands, with their options, are listed in Table 12-1.

Table 12-1: Commonfile andod Options

Commands	**Execution**
file	Examines the first few lines of a file to determine a classification
-f *filename*	Reads the list of filenames to be examined from a file
od	Prints the contents of a file byte by byte in either octal, character, decimal, or hexadecimal; octal is the default
-c	Outputs character form of byte values; nonprinting characters have a corresponding character representation
-d	Outputs decimal form of byte values
-x	Outputs hexadecimal form of byte values
-o	Outputs octal form of byte values

The File Structure

Linux organizes files into a hierarchically connected set of directories. Each directory may contain either files or other directories. In this respect, directories perform two important functions. A *directory* holds files, much like files held in a file drawer, and a directory connects to other directories, much like a branch in a tree is connected to other branches. With respect to files, directories appear to operate like file drawers, with each drawer holding several files. To access files, you open a file drawer. Unlike file drawers, however, directories can contain not only files, but other directories as well. In this way, a directory can connect to another directory.

Because of the similarities to a tree, such a structure is often referred to as a *tree structure*. This structure could more accurately be thought of as an upside-down bush rather than a tree, however, because no trunk exists. The tree is represented upside down, with the root at the top. Extending down from the root are the branches. Each branch grows out of only one branch, but it can have many lower branches. In this respect, it can be said to have a *parent-child structure*. In the same way, each directory is itself a subdirectory of one other directory. Each directory may contain many subdirectories, but is itself the child of only one parent directory.

The Linux file structure branches into several directories beginning with a root directory, /. Within the root directory several system directories contain files and programs that are features of the Linux system. The root directory also contains a directory called **home** that contains the home directories of all the users in the system. Each user's home directory, in turn, contains the directories the user has made for his or her use. Each of these could also contain directories. Such nested directories would branch out from the user's home directory, as shown in Figure 12-2.

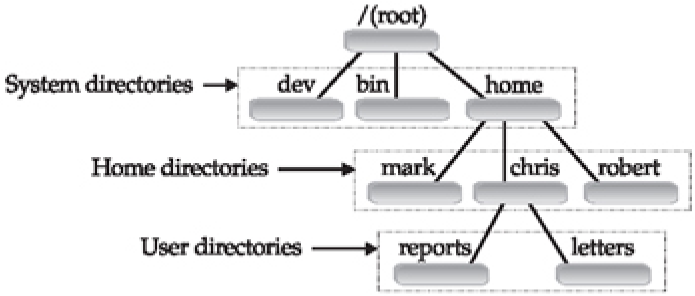


Figure 12-2: The Linux file structure beginning at the root directory

Note The user's home directory can be any directory, though it is usually the directory that bears the user's login name. This directory is located in the directory named **/home** on your Linux system. For example, a user named **dylan** will have a home directory called **dylan** located in the system's **/home** directory. The user's home directory is a subdirectory of the directory called **/home** on your system.

Home Directories

When you log into the system, you are placed within your home directory. The name given to this directory by the system is the same as your login name. Any files you create when you first log in are organized within your home directory. Within your home directory, however, you can create more directories. You can then change to these directories and store files in them. The same is true for other users on the system. Each user has his or her own home directory, identified by the appropriate login name. Users, in turn, can create their own directories.

You can access a directory either through its name or by making it the default directory. Each directory is given a name when it is created. You can use this name in file operations to access files in that directory. You can also make the directory your default directory. If you do not use any directory names in a file operation, the default directory will be accessed. The default directory is referred to as the *working directory*. In this sense, the working directory is the one from which you are currently working.

When you log in, the working directory is your home directory, usually having the same name as your login name. You can change the working directory by using the **cd** command to designate another directory as the working directory. As the working directory is changed, you can move from one directory to another. Another way to think of a directory is as a corridor. In such a corridor, there are doors with names on them. Some doors lead to rooms; others lead to other corridors. The doors that open to rooms are like files in a directory. The doors that lead to other corridors are like other directories. Moving from one corridor to the next corridor is like changing the working directory. Moving through several corridors is like moving through several directories.

Pathnames

The name you give to a directory or file when you create it is not its full name. The full name of a directory is its *pathname*. The hierarchically nested relationship among directories forms paths, and these paths can be used to identify and reference any directory or file unambiguously. In Figure 12-3, a path exists from the root directory, /, through the **home** directory to the **robert** directory. Another path exists from the root directory through the **home** and **chris** directories to the **reports** directory. Although parts of each path may at first be shared, at some point they differ. Both the directories **robert** and **reports** share the two directories, **root** and **home**. Then they differ. In the **home** directory, **robert** ends with **robert**, but the directory **chris** then leads to **reports**. In this way, each directory in the file structure can be said to have its own unique path. The actual name by which the system identifies a directory always begins with the root directory and consists of all directories nested above that directory.

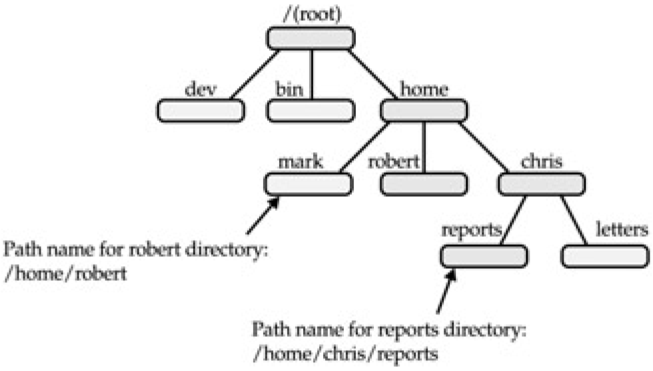


Figure 12-3: Directory pathnames

In Linux, you write a pathname by listing each directory in the path separated by a forward slash. A slash preceding the first directory in the path represents the root. The pathname for the **robert** directory is **/home/robert**. The pathname for the **reports** directory is **/home/chris/reports**. Pathnames also apply to files. When you create a file within a directory, you give the file a name. The actual name by which the system identifies the file, however, is the filename combined with the path of directories from the root tothe file's directory. In Figure 12-4, the path for the **weather** file consists of the root, **home**, and **chris** directories and the filename **weather**. The pathname for **weather** is **/home/ chris/ weather** (the root directory is represented by the first slash).

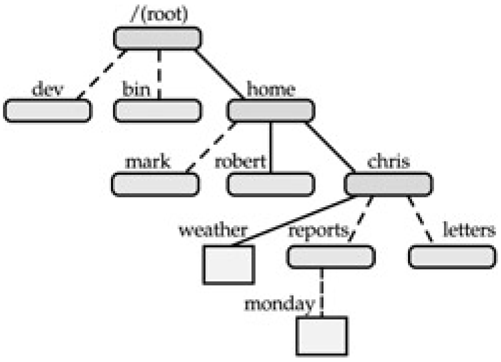


Figure 12-4: Pathname for Weather: /home/chris/weather

Pathnames may be absolute or relative. An *absolute pathname* is the complete pathname of a file or directory beginning with the root directory. A *relative pathname* begins from your working directory; it is the path of a file relative to your working directory. The working directory is the one you are currently operating in. Using the directory structure described in Figure 12-4, if **chris** is your working directory, the relative pathname for the file **monday** is **reports/monday**. The absolute pathname for **monday** is **/home/chris/ reports/monday**.

The absolute pathname from the root to your home directory could be especially complex and, at times, even subject to change by the system administrator. To make it easier to reference, you can use a special character, the tilde ~, which represents the absolute pathname of your home directory. In the next example, from the **thankyou** directory, the user references the **weather** file in the home directory by placing a tilde and slash before **weather**:

```
$ pwd
/home/chris/letters/thankyou
$ cat ~/weather
raining and warm
$
```

You must specify the rest of the path from your home directory. In the next example, the user references the **monday** file in the **reports** directory. The tilde represents the path to the user's home directory, **/home/chris**, and then the rest of the path to the **monday** file is specified.

```
$ cat ~/reports/monday
```

System Directories

The root directory that begins the Linux file structure contains several system directories. The system directories contain files and programs used to run and maintain the system. Many contain other subdirectories with programs for executing specific features of Linux. For example, the directory **/usr/bin** contains the various Linux commands that users execute, such as **cp** and **mv**. The directory **/bin** holds interfaces with different system devices, such as the printer or the terminal. Table 12-2 lists the basic system directories.

Table 12-2: Standard System Directories in Linux

Directory	**Function**
/	Begins the file system structure, called the *root*
/home	Contains users' home directories
/bin	Holds all the standard commands and utility programs
/usr	Holds those files and commands used by the system; this directory breaks down into several subdirectories
/usr/bin	Holds user-oriented commands and utility programs
/usr/sbin	Holds system administration commands
/usr/lib	Holds libraries for programming languages
/usr/share/doc	Holds Linux documentation
/usr/share/man	Holds the online manual Man files
/usr/spool	Holds spooled files, such as those generated for printing jobs and network transfers
/sbin	Holds system administration commands for booting the system

Table 12-2: Standard System Directories in Linux

Directory	**Function**
/var	Holds files that vary, such as mailbox files
/dev	Holds file interfaces for devices such as the terminals and printers
/etc	Holds system configuration files and any other system files

Note The overall organization of the Linux file structure for system directories and other useful directories such as those used for the kernel and X Window System are discussed in detail in Chapter 32.

Listing, Displaying, and Printing Files: ls, cat, more, and lpr

One of the primary functions of an operating system is the management of files. You may need to perform certain basic output operations on your files, such as displaying them on your screen or printing them. The Linux system provides a set of commands that perform basic file-management operations, such as listing, displaying, and printing files, as well as copying, renaming, and erasing files. These commands are usually made up of abbreviated versions of words. For example, the **ls** command is a shortened form of "list" and lists the files in your directory. The **lpr** command is an abbreviated form of "line print" and will print a file. The **cat** and **more** commands display the contents of a file on the screen. Table 12-3 lists these commands with their different options. When you log in to your Linux system, you may want a list of the files in your home directory. The **ls** command, which outputs a list of your file and directory names, is useful for this. The **ls** command has many possible options for displaying filenames according to specific features. These are discussed in more detail at the end of the chapter.

Table 12-3: Listing, Displaying, and Printing Files

Command or Option	**Execution**
ls	This command lists file and directory names: **$ ls** *filenames*
cat	This filter can be used to display a file. It can take filenames for its arguments. It outputs the contents of those files directly to the standard output, which, by default, is directed to the screen: **$ cat** *filenames*
more	This utility displays a file screen by screen. It can take filenames for its arguments. It outputs the contents of those files to the screen, one screen at a time: **$ more** *filenames*
more Options	
+*num*	Begins displaying the file at page *num*.
more Commands	
*Num***f**	Skips forward *num* number of screens.
*Num***b**	Skips backward *num* number of screens.
d	Displays half a screen.

Table 12-3: Listing, Displaying, and Printing Files

Command or Option	Execution
h	Lists all **more** commands.
q	Quits more utility.
lpr	Sends a file to the line printer to be printed; a list of files may be used as arguments: **$ lpr** *filenames*
lpr Options	
-P *printer-name*	Selects a specific printer.
lpq	Lists the print queue for printing jobs.
lprm	Removes a printing job from the printing queue.

Displaying Files: cat and more

You may also need to look at the contents of a file. The **cat** and **more** commands display the contents of a file on the screen. "cat" stands for *concatenate.* The **cat** command is complex and versatile, as described in Chapter 11. Here it is used in a limited way, displaying the text of a file on the screen:

```
$ cat mydata
computers
```

The **cat** command outputs the entire text of a file to the screen at once. This presents a problem when the file is large because its text quickly speeds past on the screen. The **more** command is designed to overcome this limitation by displaying one screen of text at a time. You can then move forward or backward in the text at your leisure. You invoke the **more** command by entering the command name followed by the name of the file you want to view.

```
$ more mydata
```

When **more** invokes a file, the first screen of text is displayed. To continue to the next screen, you press the F key or the SPACEBAR. To move back in the text, you press the B key. You can quit at any time by pressing the Q key.

Printing Files: lpr, lpq, and lprm

With the printer commands like **lpr** and **lprm**, you can perform printing operations like printing files or canceling print jobs (see Table 12-3). When you need to print files, use the **lpr** command to send files to the printer connected to your system. In the next example, the user prints the **mydata** file:

```
$ lpr mydata
```

If you want to print several files at once, you can specify more than one file on the command line after the **lpr** command. In the next example, the user prints out both the **mydata** and **preface** files:

```
$ lpr mydata preface
```

Printing jobs are placed in a queue and printed one at a time in the background. You can continue with other work as your files print. You can see the position of a particular printing job at any given time with the **lpq** command. **lpq** gives the owner of the printing job (the login name of the user who sent the job), the print job ID, the size in bytes, and the temporary file in which it is currently held. In this example, the owner is **chris** and the print ID is **00015**:

```
$ lpq
Owner ID Chars Filename
chris 00015 360 /usr/lpd/cfa00015
```

If you need to cancel an unwanted printing job, you can do so with the **lprm** command. **lprm** takes as its argument either the ID number of the printing job or the owner's name. **lprm** then removes the print job from the print queue. For this task, **lpq** is helpful, for it provides you with the ID number and owner of the printing job you need to use with **lprm**. In the next example, the print job 15 is canceled:

```
$ lprm 00015
```

You can have several printers connected to your Linux system. One of these will be designated the default printer, and **lpr** prints to this printer unless another printer is specified. With **lpr**, you can specify the particular printer on which you want your file printed. Each printer on your system will have its own name. You can specify which printer to use with the **-P** option followed by that printer's name. In the next example, the file **mydata** is printed on the evans1 printer:

```
$ lpr -Pevans1 mydata
```

Managing Directories: mkdir, rmdir, ls, cd, and pwd

You can create and remove your own directories, as well as change your working directory, with the **mkdir**, **rmdir**, and **cd** commands. Each of these commands can take as its argument the pathname for a directory. The **pwd** command displays the absolute pathname of your working directory. In addition to these commands, the special characters represented by a single dot, a double dot, and a tilde can be used to reference the working directory, the parent of the working directory, and the home directory, respectively. Taken together, these commands enable you to manage your directories. You can create nested directories, move from one directory to another, and use pathnames to reference any of your directories. Those commands commonly used to manage directories are listed in Table 12-4.

Table 12-4: Directory Commands

Command	**Execution**
mkdir	Creates a directory: **$ mkdir reports**
rmdir	Erases a directory: **$ rmdir letters**
ls -F	Lists directory name with a preceding slash: **$ ls -F** **today /reports /letters**
ls -R	Lists working directory as well as all subdirectories.

Table 12-4: Directory Commands

Command	Execution
cd *directory name*	Changes to the specified directory, making it the working directory. **cd** without a directory name changes back to the home directory: **$ cd reports** **$ cd**
pwd	Displays the pathname of the working directory: **$ pwd** **/home/chris/reports**
directory name/filename	A slash is used in pathnames to separate each directory name. In the case of pathnames for files, a slash separates the preceding directory names from the filename: **$ cd /home/chris/reports** **$ cat /home/chris/reports/mydata**
..	References the parent directory. You can use it as an argument or as part of a pathname: **$ cd ..** **$ mv ../larisa oldletters**
.	References the working directory. You can use it as an argument or as part of a pathname: **$ ls .** **$ mv ../aleina .**
~/pathname	The tilde is a special character that represents the pathname for the home directory. It is useful when you need to use an absolute pathname for a file or directory: **$ cp monday ~/today** **$ mv tuesday ~/weather**

You create and remove directories with the **mkdir** and **rmdir** commands. In either case, you can also use pathnames for the directories. In the next example, the user creates the directory **reports**. Then, the user creates the directory **letters** using a pathname.

```
$ mkdir reports
$ mkdir /home/chris/letters
```

You can remove a directory with the **rmdir** command followed by the directory name. In the next example, the user removes the directory **reports** with the **rmdir** command. Then, the directory **letters** is removed using its pathname.

```
$ rmdir reports
$ rmdir /home/chris/letters
```

You have seen how to use the **ls** command to list the files and directories within your working directory. To distinguish between file and directory names, however, you need to use the **ls** command with the **-F** option. A slash is then placed after each directory name in the list.

```
$ ls
weather reports letters
```

```
$ ls -F
weather reports/ letters/
```

The **ls** command also takes as an argument any directory name or directory pathname. This enables you to list the files in any directory without first having to change to that directory. In the next example, the **ls** command takes as its argument the name of a directory, **reports**. Then the **ls** command is executed again, only this time the absolute pathname of **reports** is used.

```
$ ls reports
monday tuesday
$ ls /home/chris/reports
monday tuesday
$
```

Within each directory, you can create still other directories; in effect, nesting directories. Using the **cd** command, you can change from one directory to another. No indicator tells you what directory you are currently in, however. To find out what directory you have changed to, use the **pwd** command to display the name of your current working directory. The **pwd** command displays more than just the name of the directory-it displays the full pathname, as shown in the next example. The pathname displayed here consists of the user's home directory, **dylan**, and the directory it is a part of, the directory called **home**, and directory that **home** is part of, /, the root directory. Each directory name is separated by a slash. The root directory is represented by a beginning slash.

```
$ pwd
/home/dylan
```

As you already know, you can change directories with the **cd** command. Changing to a directory makes that directory the working directory, which is your default directory.

Note File commands, such as **ls** and **cp**, unless specifically told otherwise, operate on files in your working directory.

When you log into the system, your working directory is your home directory. When a user account is created, the system also creates a home directory for that user. When you log in, you are always placed in your home directory. The **cd** command enables you to make another directory the working directory. In a sense, you can move from your home directory into another directory. This other directory then becomes the default directory for any commands and any new files created. For example, the **ls** command now lists files in this new working directory.

The **cd** command takes as its argument the name of the directory to which you want to change. The name of the directory can be the name of a subdirectory in your working directory or the full pathname of any directory on the system. If you want to change back to your home directory, you only need to enter the **cd** command by itself, without a filename argument.

```
$ pwd
/home/dylan
$ cd props
$ pwd
/home/dylan/props
```

```
$ cd /home/chris/letters
$ pwd
/home/chris/letters
$
```

You can use a double dot, **..**, to represent a directory's parent. This literally represents the pathname of the parent directory. You can use the double dot symbol with the **cd** command to move back up to the parent directory, making the parent directory the current directory. In the next example, the user moves to the **letters** directory and then changes back to the home directory:

```
$ cd letters
$ pwd
/home/chris/letters
$ cd ..
$ pwd
/home/chris
```

A directory always has a parent (except, of course, for the root). For example, in the previous listing, the parent for **thankyou** is the **letters** directory. When a directory is created, two entries are made: one represented with a dot, **.**, and the other represented by a double dot, **..** . The dot represents the pathnames of the directory, and the double dot represents the pathname of its parent directory. The double dot, used as an argument in a command, references a parent directory. The single dot references the directory itself. In the next example, the user changes to the **letters** directory. The **ls** command is used with the **.** argument to list the files in the **letters** directory. Then, the **ls** command is used with the **..** argument to list the files in the parent directory of **letters**, the **chris** directory.

```
$ cd letters
$ ls .
thankyou
$ ls ..
weather letters
$
```

You can use the single dot to reference your working directory, instead of using its pathname. For example, to copy a file to the working directory retaining the same name, the dot can be used in place of the working directory's pathname. In this sense, the dot is another name for the working directory. In the next example, the user copies the **weather** file from the **chris** directory to the **reports** directory. The **reports** directory is the working directory and can be represented with the single dot.

```
$ cd reports
$ cp /home/chris/weather .
```

The **..** symbol is often used to reference files in the parent directory. In the next example, the **cat** command displays the **weather** file in the parent directory. The pathname for the file is the **..** symbol followed by a slash and the filename.

```
$ cat ../weather
raining and warm
```

Note You can use the **cd** command with the **..** symbol to step back through successive parent directories of the directory tree from a lower directory.

File and Directory Operations: find, cp, mv, rm, and ln

As you create more and more files, you may want to back them up, change their names, erase some of them, or even give them added names. Linux provides you with several file commands that enable you to search for files, copy files, rename files, or remove files (see Tables 12-5 and 12-6). If you have a large number of files, you can also search them to locate a specific one. The commands are shortened forms of full words, consisting of only two characters. The **cp** command stands for "copy" and copies a file, **mv** stands for "move" and renames or moves a file, **rm** stands for "remove" and erases a file, and **ln** stands for "link" and adds another name for a file. One exception to this rule is the **find** command, which performs searches of your filenames to find a file.

Table 12-5: Thefind Command

Command or Option	**Execution**
find	Searches directories for files based on search criteria. This command has several options that specify the type of criteria and actions to be taken.
-name *pattern*	Searches for files with the *pattern* in the name.
-group *name*	Searches for files belonging to this group *name.*
-size *num***c**	Searches for files with the size *num* in blocks. If **c** is added after *num*, then the size in bytes (characters) is searched for.
-mtime *num*	Searches for files last modified *num* days ago.
-newer *pattern*	Searches for files modified after the one matched by *pattern.*
-print	Outputs the result of the search to the standard output. The result is usually a list of filenames, including their full pathnames.
-type *filetype*	Searches for files with the specified file type.
b	Block device file.
c	Character device file.
d	Directory file.
f	Ordinary (regular) file.
p	Named pipes (fifo).
l	Symbolic links.

Table 12-6: File Operations

Command	**Execution**
cp *filename filename*	Copies a file. **cp** takes two arguments: the original file and the name of the new copy. You can use pathnames for the files to copy across directories: **$ cp today reports/monday**
cp -r *dirname dirname*	Copies a subdirectory from one directory to another. The copied directory includes all its own subdirectories: **$ cp -r letters/thankyou oldletters**

Table 12-6: File Operations

Command	**Execution**
mv *filename filename*	Moves (renames) a file. **mv** takes two arguments: the first is the file to be moved. The second argument can be the new filename or the pathname of a directory. If it is the name of a directory, then the file is literally moved to that directory, changing the file's pathname: **$ mv today /home/chris/reports**
mv *dirname dirname*	Moves directories. In this case, the first and last arguments are directories: **$ mv letters/thankyou oldletters**
ln *filename filename*	Creates added names for files referred to as links. A link can be created in one directory that references a file in another directory: **$ ln today reports/monday**
rm *filenames*	Removes (erases) a file. Can take any number of filenames as its arguments. Literally removes links to a file. If a file has more than one link, you need to remove all of them to erase a file: **$rm today weather weekend**

Searching Directories: find

Once you have a large number of files in many different directories, you may need to search them to locate a specific file, or files, of a certain type. The **find** command enables you to perform such a search. The **find** command takes as its arguments directory names followed by several possible options that specify the type of search and the criteria for the search. **find** then searches within the directories listed and their subdirectories for files that meet these criteria. The **find** command can search for a file based on its name, type, owner, and even the time of the last update.

```
$ find directory-list -option criteria
```

The **-name** option has as its criteria a pattern and instructs **find** to search for the filename that matches that pattern. To search for a file by name, you use the **find** command with the directory name followed by the **-name** option and the name of the file.

```
$ find directory-list -name filename
```

The **find** command also has options that merely perform actions, such as outputting the results of a search. If you want **find** to display the filenames it has found, you simply include the **-print** option on the command line along with any other options. The **-print** option instructs **find** to output to the standard output the names of all the files it locates. In the next example, the user searches for all the files in the **reports** directory with the name **monday**. Once located, the file, with its relative pathname, is printed.

```
$ find reports -name monday -print
reports/monday
```

The **find** command prints out the filenames using the directory name specified in the directory list. If you specify an absolute pathname, the absolute path of the found directories will be output. If you specify a relative pathname, only the relative pathname is output. In the previous example, the user specified a relative pathname, **reports**, in the directory list. Located filenames were output beginning with this relative pathname. In the next example, the user specifies an absolute pathname in the directory list. Located filenames are then output using this absolute pathname.

```
$ find /home/chris -name monday -print
/home/chris/reports/monday
```

If you want to search your working directory, you can use the dot in the directory pathname to represent your working directory. The double dot would represent the parent directory. The next example searches all files and subdirectories in the working directory, using the dot to represent the working directory. If you are located in your home directory, this is a convenient way to search through all your own directories. Notice the located filenames are output beginning with a dot.

```
$ find . -name weather -print
./weather
```

You can use shell wildcard characters as part of the pattern criteria for searching files. The special character must be quoted, however, to avoid evaluation by the shell. In the next example, all files with the **.c** extension in the **programs** directory are searched for:

```
$ find programs -name '*.c' -print
```

You can also use the **find** command to locate other directories. In Linux, a directory is officially classified as a special type of file. Although all files have a byte-stream format, some files, such as directories, are used in special ways. In this sense, a file can be said to have a file type. The **find** command has an option called **-type** that searches for a file of a given type. The **-type** option takes a one-character modifier that represents the file type. The modifier that represents a directory is a **d**. In the next example, both the directory name and the directory file type are used to search for the directory called **thankyou**:

```
$ find /home/chris -name thankyou -type d -print
/home/chris/letters/thankyou
$
```

File types are not so much different types of files as they are the file format applied to other components of the operating system, such as devices. In this sense, a device is treated as a type of file, and you can use **find** to search for devices and directories, as well as ordinary files. Table 12-5 lists the different types available for the **find** command's **-type** option.

Moving and Copying Files

To make a copy of a file, you simply give **cp** two filenames as its arguments (see Table 12-6). The first filename is the name of the file to be copied-the one that already exists. This is often referred to as the *source file.* The second filename is the name you want for the copy. This will be a new file containing a copy of all the data in the source file. This second argument is often referred to as the *destination file.* The syntax for the **cp** command follows:

```
$ cp source-file destination-file
```

In the next example, the user copies a file called **proposal** to a new file called **oldprop**:

```
$ cp proposal oldprop
```

When the user lists the files in that directory, the new copy will be among them.

```
$ ls
proposal oldprop
```

You could unintentionally destroy another file with the **cp** command. The **cp** command generates a copy by first creating a file and then copying data into it. If another file has the same name as the destination file, then that file is destroyed and a new file with that name is created. In a sense, the original file is overwritten with the new copy. In the next example, the **proposal** file is overwritten by the **newprop** file. The **proposal** file already exists.

```
$ cp newprop proposal
```

Most Linux distributions configure your system to detect this overwrite condition. If not, you can use the **cp** command with the **-i** (for interactive) option to detect it. With this option, **cp** first checks to see if the file already exists. If it does, you are then asked if you want to overwrite the existing file. If you enter **y**, the existing file is destroyed and a new one created as the copy. If you enter anything else, this is taken as a negative answer and the **cp** command is interrupted, preserving the original file.

```
$ cp -i newprop proposal
Overwrite proposal? n
$
```

To copy a file from your working directory to another directory, you only need to use that directory name as the second argument in the **cp** command. The name of the new copy will be the same as the original, but the copy will be placed in a different directory. Files in different directories can have the same names. Because files are in different directories, they are registered as different files.

```
$ cp filenames directory-name
```

The **cp** command can take a list of several filenames for its arguments, so you can copy more than one file at a time to a directory. Simply specify the filenames on the command line, entering the directory name as the last argument. All the files are then copied to the specified directory. In the next example, the user copies both the files **preface** and **doc1** to the **props** directory. Notice **props** is the last argument.

```
$ cp preface doc1 props
```

You can use any of the wildcard characters to generate a list of filenames to use with **cp** or **mv**. For example, suppose you need to copy all your C source code files to a given directory. Instead of listing each one individually on the command line, you could use a * character with the **.c** extension to match on and generate a list of C source code files (all files with a **.c** extension). In the next example, the user copies all source code files in the current directory to the **sourcebks** directory:

```
$ cp *.c sourcebks
```

If you want to copy all the files in a given directory to another directory, you could use * to match on and generate a list of all those files in a **cp** command. In the next example, the user copies all the files in the **props** directory to the **oldprop** directory. Notice the use of a **props** pathname preceding the * special characters. In this context, **props** is a pathname that will be appended before each file in the list that * generates.

```
$ cp props/* oldprop
```

You can, of course, use any of the other special characters, such as **.**, **?**, or **[]**. In the next example, the user copies both source code and object code files (**.c** and **.o**) to the **projbk** directory:

```
$ cp *.[oc] projbk
```

When you copy a file, you may want to give the copy a different name than the original. To do so, place the new filename after the directory name, separated by a slash.

```
$ cp filename directory-name/new-filename
```

In the next example, the file **newprop** is copied to the directory **props** and the copy is given the name **version1**. The user then changes to the **props** directory and lists the files. Only one file exists, which is called **version1**.

```
$ cp newprop props/version1
$ cd props
$ ls
version1
```

You can use the **mv** command either to rename a file or to move a file from one directory to another. When using **mv** to rename a file, you simply use the new filename as the second argument. The first argument is the current name of the file you are renaming.

```
$ mv original-filename new-filename
```

In the next example, the **proposal** file is renamed with the name **version1**:

```
$ mv proposal version1
```

As with **cp**, it is easy for **mv** to erase a file accidentally. When renaming a file, you might accidentally choose a filename already used by another file. In this case, that other file will be erased. The **mv** command also has an **-i** option that checks first to see if a file by that name already exists. If it does, then you are asked first if you want to overwrite it. In the next example, a file already exists with the name **version1**. The overwrite condition is detected and you are asked whether you want to overwrite that file.

```
$ ls
proposal version1
$ mv -i version1 proposal
Overwrite proposal? n
$
```

You can move a file from one directory to another by using the directory name as the second argument in the **mv** command. In this case, you can think of the **mv** command as simply moving a file from one directory to another, rather than renaming the file. After you move the file, it will have the same name as it had in its original directory unless you specify otherwise.

```
$ mv filename directory-name
```

If you want to rename a file when you move it, you can specify the new name of the file after the directory name. The directory name and the new filename are separated by a forward slash. In the next example, the file **newprop** is moved to the directory **props** and renamed as **version1**:

```
$ mv newprops props/version1
$ cd props
$ ls
version1
```

You can also use any of the special characters described in Chapter 5 to generate a list of filenames to use with **mv**. In the next example, the user moves all source code files in the current directory to the **newproj** directory:

```
$ mv *.c newproj
```

If you want to move all the files in a given directory to another directory, you can use * to match on and generate a list of all those files. In the next example, the user moves all the files in the **reports** directory to the **repbks** directory:

```
$ mv reports/* repbks
```

Moving and Copying Directories

You can also copy or move whole directories at once. Both **cp** and **mv** can take as their first argument a directory name, enabling you to copy or move subdirectories from one directory into another (see Table 12-6). The first argument is the name of the directory to be moved or copied, while the second argument is the name of the directory within which it is to be placed. The same pathname structure used for files applies to moving or copying directories.

You can just as easily copy subdirectories from one directory to another. To copy a directory, the **cp** command requires you to use the **-r** option. The **-r** option stands for "recursive." It directs the **cp** command to copy a directory, as well as any subdirectories it may contain. In other words, the entire directory subtree, from that directory on, will be copied. In the next example, the **thankyou** directory is copied to the **oldletters** directory. Now two **thankyou** subdirectories exist, one in **letters** and one in **oldletters**.

```
$ cp -r letters/thankyou oldletters
$ ls -F letters
/thankyou
$ ls -F oldletters
/thankyou
```

Erasing a File: the rm Command

As you use Linux, you will find the number of files you use increases rapidly. Generating files in Linux is easy. Applications such as editors, and commands such as **cp**, easily create files. Eventually, many of these files may become outdated and useless. You can then remove them with the **rm** command. In the next example, the user erases the file **oldprop**:

```
$ rm oldprop
```

The **rm** command can take any number of arguments, enabling you to list several filenames and erase them all at the same time. You just list them on the command line after you type **rm**.

```
$ rm proposal version1 version2
```

Be careful when using the **rm** command, because it is irrevocable. Once a file is removed, it cannot be restored (there is no undo). Suppose, for example, you enter the **rm** command by accident while meaning to enter some other command, such as **cp** or **mv**. By the time you press ENTER and realize your mistake, it is too late. The files are gone. To protect against this kind of situation, you can use the **rm** command's **-i** option to confirm you want to erase a file. With the **-i** option, you are prompted separately for each file and asked whether to remove it. If you enter **y**, the file will be removed. If you enter anything else, the file is not removed. In the next example, the **rm** command is instructed to erase the files **proposal** and **oldprop**. The **rm** command then asks for confirmation for each file. The user decides to remove **oldprop,** but not **proposal**.

```
$ rm -i proposal oldprop
Remove proposal? n
Remove oldprop? y
$
```

To remove a directory and all its subdirectories you use the **rm** command with the **-r** option. This is a very powerful command and could easily be used to erase all your files. The following example deletes the **reports** directory and all its subdirectories:

```
rm -r reports
```

Be careful of using the asterisk matching character, as described in Chapter 11. The following command will erase every file in your current working directory.

```
rm *
```

Note This is a very powerful operation, capable of erasing large segments of your file systems. Use with caution.

Links: the ln Command

You can give a file more than one name using the **ln** command. You might want to reference a file using different filenames to access it from different directories. The added names are often referred to as *links.*

The **ln** command takes two arguments: the name of the original file and the new, added filename. The **ls** operation lists both filenames, but only one physical file will exist.

```
$ ln original-file-name added-file-name
```

In the next example, the **today** file is given the additional name **weather**. It is just another name for the **today** file.

```
$ ls
today
$ ln today weather
$ ls
today weather
```

You can give the same file several names by using the **ln** command on the same file many times. In the next example, the file **today** is given both the name **weather** and **weekend**:

```
$ ln today weather
$ ln today weekend
$ ls
today weather weekend
```

You can use the **ls** command with the **-l** option to find if a file has several links. **ls** with **-l** lists several pieces of information, such as permissions, the number of links a file has, its size, and the date it was last modified. In this line of information, the first number, which precedes the user's login name, specifies the number of links a file has. The number before the date is the size of the file. The date is the last time a file was modified. In the next example, the user lists the full information for both **today** and **weather**. Notice the number of links in both files is two. Furthermore, the size and date are the same. This suggests both files are actually different names for the same file.

```
$ ls -l today weather
-rw-rw-r-- 2 chris group 563 Feb 14 10:30 today
-rw-rw-r-- 2 chris group 563 Feb 14 10:30 weather
```

This still does not tell you specifically what filenames are linked. You can be somewhat sure if two files have exactly the same number of links, sizes, and modification dates, as in the case of the files **today** and **weather**. To be certain, however, you can use the **ls** command with the **-i** option. With the **-i** option, the **ls** command lists the filename and its inode number. An *inode* number is a unique number used by the system to identify a specific file. If two filenames have the same inode number, they reference exactly the same file. They are two names for the same file. In the next example, the user lists **today**, **weather**, and **larisa**. Notice that **today** and **weather** have the same inode number.

```
$ ls -i today weather larisa
1234 today 1234 weather 3976 larisa
```

The added names, or links, created with **ln** are often used to reference the same file from different directories. A file in one directory can be linked to and accessed from another directory. Suppose you need to reference a file in the home directory from within another directory. You can set up a link from that directory to the file in the home directory. This link is actually another name for the file. Because the link is in another directory, it can have the same name as the original file.

To link a file in the home directory to another directory, use the name of that directory as the second argument in the **ln** command.

```
$ ln filename directory-name
```

In the next example, the file **today** in the **chris** directory is linked to the **reports** directory. The **ls** command lists the **today** file in both the **chris** directory and the **reports** directory. In fact, only one copy of the **today** file exists, the original file in the home directory.

```
$ ln today reports
$ ls
today reports
$ ls reports
today
$
```

Just as with the **cp** and **mv** commands, you can give another name to the link. Simply place the new name after the directory name, separated by a slash. In the next example, the file **today** is linked to the **reports** directory with the name **wednesday**. Only one actual file still exists, the original file called **today** in the **chris** directory. However, **today** is now linked to the directory **reports** with the name **wednesday**. In this sense, **today** has been given another name. In the **reports** directory, the **today** file goes by the name **wednesday**.

```
$ ln today reports/wednesday
$ ls
today reports
$ ls reports
wednesday
$
```

You can easily link a file in any directory to a file in another directory by referencing the files with their pathnames. In the next example, the file **monday** in the **reports** directory is linked to the directory **chris**. Notice the second argument is an absolute pathname.

```
$ ln monday /home/chris
```

To erase a file, you need to remove all its links. The name of a file is actually considered a link to that file-hence the command **rm** that removes the link to the file. If you have several links to the file and remove only one of them, the others stay in place and you can reference the file through them. The same is true even if you remove the original link-the original name of the file. Any added links will work just as well. In the next example, the **today** file is removed with the **rm** command. However, a link to that same file exists, called **weather**. The file can then be referenced under the name **weather**.

```
$ ln today weather
$ rm today
$ cat weather
The storm broke today
and the sun came out.
$
```

Symbolic Links and Hard Links

Linux supports what are known as symbolic links. Links, as they have been described so far, are called *hard links*. Although hard links will suffice for most of your needs, they suffer from one major limitation. A hard link may in some situations fail when you try to link to a file on some other user's directory. This is because the Linux file structure can be physically segmented into what are called file systems. A *file system* can be made up of any physical memory device or devices, from a floppy disk to a bank of hard disks. Although the files and directories in all file systems are attached to the same overall directory tree, each file system physically manages its own files and directories. This means a file in one file system cannot be linked by a hard link to a file in another file system. If you try to link to a file on another user's directory that is located on another file system, your hard link will fail. Another consideration is that there are security issues with hard links, where a hard link could be used to access a secure area.

To overcome this restriction, you use symbolic links. A *symbolic link* holds the pathname of the file to which it is linking. It is not a direct hard link but, rather, information on how to locate a specific file. Instead of registering another name for the same file as a hard link does, a symbolic link can be thought of as another symbol that represents the file's pathname. A symbolic link is another way of writing the file's pathname.

You create a symbolic link using the **ln** command with the **-s** option. In the next example, the user creates a link called **lunch** to the file **/home/george/veglist**:

```
$ ln -s lunch /home/george/veglist
```

If you list the full information about a symbolic link and its file, you will find the information displayed is different. In the next example, the user lists the full information for both **lunch** and **/home/george/veglist** using the **ls** command with the **-l** option. The first character in the line specifies the file type. Symbolic links have their own file type represented by a **l**. The file type for **lunch** is **l**, indicating it is a symbolic link, not an ordinary file. The number after the term "group" is the size of the file. Notice the sizes differ. The size of the **lunch** file is only 4 bytes. This is because **lunch** is only a symbolic link-a file that holds the pathname of another file-and a pathname takes up only a few bytes. It is not a direct hard link to the **veglist** file.

```
$ ls lunch /home/george/veglist
lrw-rw-r-- 1 chris group 4 Feb 14 10:30 lunch
-rw-rw-r-- 1 george group 793 Feb 14 10:30 veglist
```

To erase a file, you need to remove only its hard links. If any symbolic links are left over, they will be unable to access the file. In this case, a symbolic link would hold the pathname of a file that no longer exists.

Unlike hard links, you can use symbolic links to create links to directories. In effect, you can create another name with which you can reference a directory. If you use a symbolic link for a directory name, however, remember the **pwd** command always displays the actual directory name, not the symbolic name. In the next example, the user links the directory **thankyou** with the symbolic link **gifts**. When the user uses **gifts** in the **cd** command, the user is actually changed to the **thankyou** directory. **pwd** displays the pathname for the **thankyou** directory.

```
$ ln -s /home/chris/letters/thankyou gifts
$ cd gifts
$ pwd
/home/chris/letters/thankyou
$
```

If you want to display the name of the symbolic link, you can access it in the **cwd** variable. The **cwd** variable is a special system variable that holds the name of a directory's symbolic link, if one exists. Variables such as **cwd** are discussed in Chapter 13. You display the contents of **cwd** with the command **echo $cwd**.

```
$ pwd
/home/chris/letters/thankyou
$ echo $cwd
/home/chris/gifts
```

File and Directory Permissions: chmod

Each file and directory in Linux contains a set of permissions that determine who can access them and how. You set these permissions to limit access in one of three ways: You can restrict access to yourself alone, you can allow users in a predesignated group to have access, or you can permit anyone on your system to have access; and, you can control how a given file or directory is accessed. A file and directory may have read, write, and execute permissions. When a file is created, it is automatically given read and write permissions for the owner, enabling you to display and modify the file. You may change these permissions to any combination you want. A file could have read-only permission, preventing any modifications. A file could also have execute permission, allowing it to be executed as a program.

Three different categories of users can have access to a file or directory: the owner, the group, or others. The owner is the user who created the file. Any file you create, you own. You can also permit your group to have access to a file. Often, users are collected into groups. For example, all the users for a given class or project could be formed into a group by the system administrator. A user can give access to a file to other members of the group. Finally, you can also open up access to a file to all other users on the system. In this case, every user on your system could have access to one of your files or directories. In this sense, every other user on the system makes up the "others" category.

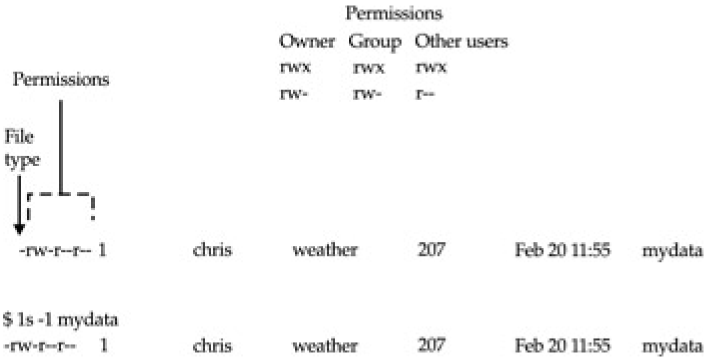
Each category has its own set of read, write, and execute permissions. The first set controls the user's own access to his or her files-the owner access. The second set controls the access of the group to a user's files. The third set controls the access of all other users to the user's files. The three sets of read, write, and execute permissions for the three categories-owner, group, and other-make a total of nine types of permissions.

As you saw in the previous section, the **ls** command with the **-l** option displays detailed information about the file, including the permissions. In the next example, the first set of characters on the left is a list of the permissions set for the **mydata** file:

```
$ ls -l mydata
-rw-r--r-- 1 chris weather 207 Feb 20 11:55 mydata
```

An empty permission is represented by a dash, **-**. The read permission is represented by *r*, write by *w*, and execute by *x*. Notice there are ten positions. The first character indicates the file type. In a general sense, a directory can be considered a type of file. If the first character is a dash, a file is being listed. If it is *d,* information about a directory is being displayed.

The next nine characters are arranged according to the different user categories. The first set of three characters is the owner's set of permissions for the file. The second set of three characters is the group's set of permissions for the file. The last set of three characters is the other users' set of permissions for the file. In Figure 12-5, the **mydata** file has the read and write permissions set for the owner category, the read permission only set for the group category, and the read permission set for the other users category. This means, although anyone in the group or any other user on the system can read the file, only the owner can modify it.



```
$ ls -l mydata
-rw-r--r--   1        chris       weather      207         Feb 20 11:55     mydata
```

Figure 12-5: File permissions

You use the **chmod** command to change different permission configurations. **chmod** takes two lists as its arguments: permission changes and filenames. You can specify the list of permissions in two different ways. One way uses permission symbols and is referred to as the *symbolic method.* The other uses what is known as a "binary mask" and is referred to as either the *absolute* or the *relative method.* Of the two, the symbolic method is the more intuitive and will be presented first. Table 12-7 lists options for the **chmod** command.

Table 12-7: File and Directory Permission Operations

Command or Option	**Execution**
chmod	Changes the permission of a file or directory.
Options	
>+	Adds a permission.
>**-**	Removes a permission.
>=	Assigns entire set of permissions.
>**r**	Sets read permission for a file or directory. A file can be displayed or printed. A directory can have the list of its files displayed.
>**w**	Sets write permission for a file or directory. A file can be edited or erased. A directory can be removed.
>**x**	Sets execute permission for a file or directory. If the file is a shell script, it can be executed as a program. A directory can be changed to and entered.
>**u**	Sets permissions for the user who created and owns the file or

Table 12-7: File and Directory Permission Operations

Command or Option	Execution
	directory.
>**g**	Sets permissions for group access to a file or directory.
>**o**	Sets permissions for access to a file or directory by all other users on the system.
>**a**	Sets permissions for access by the user, group, and all other users.
>**s**	Sets User ID and Group ID permission; program owned by owner and group.
>**t**	Sets sticky bit permission; program remains in memory.
chgrp *groupname filenames*	Changes the group for a file or files.
chown *user-name filenames*	Changes the owner of a file or files.
ls -l *filename*	Lists a filename with its permissions displayed.
ls -ld *directory*	Lists a directory name with its permissions displayed.
ls -l	Lists all files in a directory with its permissions displayed.

Setting Permissions: Permission Symbols

As you might have guessed, the symbolic method of setting permissions uses the characters *r, w,* and *x* for read, write, and execute, respectively. Any of these permissions can be added or removed. The symbol to add a permission is the plus sign, +. The symbol to remove a permission is the minus sign, **-**. In the next example, the **chmod** command adds the execute permission and removes the write permission for the **mydata** file. The read permission is not changed.

```
$ chmod +x-w mydata
```

Permission symbols also specify each user category. The owner, group, and others categories are represented by the *u, g,* and *o* characters, respectively. Notice the owner category is represented by a *u* and can be thought of as the user. The symbol for a category is placed before the read, write, and execute permissions. If no category symbol is used, all categories are assumed, and the permissions specified are set for the user, group, and others. In the next example, the first **chmod** command sets the permissions for the group to read and write. The second **chmod** command sets permissions for other users to read. Notice no spaces are between the permission specifications and the category. The permissions list is simply one long phrase, with no spaces.

```
$ chmod g+rw mydata
$ chmod o+r mydata
```

A user may remove permissions as well as add them. In the next example, the read permission is set for other users, but the write and execute permissions are removed:

```
$ chmod o+r-wx mydata
```

Another permission symbol exists, **a**, which represents all the categories. The *a* symbol is the default. In the next example, both commands are equivalent. The read permission is explicitly set with the *a* symbol denoting all types of users: other, group, and user.

```
$ chmod a+r mydata
$ chmod +r mydata
```

One of the most common permission operations is setting a file's executable permission. This is often done in the case of shell program files, which are discussed in Chapters 12 and 16. The executable permission indicates a file contains executable instructions and can be directly run by the system. In the next example, the file **lsc** has its executable permission set and then executed:

```
$ chmod u+x lsc
$ lsc
main.c lib.c
$
```

In addition to the read/write/execute permissions, you can also set ownership permissions for executable programs. Normally, the user who runs a program owns it while it is running, even though the program file itself may be owned by another user. The Set User ID permission allows the original owner of the program to own it always, even while another user is running the program. For example, most software on the system is owned by the root user, but is run by ordinary users. Some such software may have to modify files owned by the root. In this case, the ordinary user would need to run that program with the root retaining ownership so the program could have the permissions to change those root-owned files. The Group ID permission works the same way, except for groups. Programs owned by a group retain ownership, even when run by users from another group. The program can then change the owner group's files. There is a potential security risk involved in that you are essentially giving a user some limited root-level access.

To add both the User ID and Group ID permissions to a file, you use the **s** option. The following example adds the User ID permission to the **pppd** program, which is owned by the root user. When an ordinary user runs **pppd**, the root user retains ownership, allowing the **pppd** program to change root-owned files.

```
# chmod +s /usr/sbin/pppd
```

The Set User ID and Set Group ID permissions show up as an *s* in the execute position of the owner and group segments. Set User ID and Group ID are essentially variations of the execute permission, *x*. Read, write, and User ID permission would be **rws** instead of just **rwx**.

```
# ls -l /usr/sbin/pppd
-rwsr-sr-x 1 root root 84604 Aug 14 1996 /usr/sbin/pppd
```

One other special permission provides efficient use of programs. The sticky bit instructs the system to keep a program in memory (on a swap device) after it finishes execution. This is useful for small programs used frequently by many users. The sticky bit permission is *t*. The sticky bit shows up as a *t* in the execute position of the other permissions. A program with read and execute permission with the sticky bit would have its permissions displayed as **r-t**.

```
# chmod +t /usr/X11R6/bin/xtetris
# ls -l /usr/X11R6/bin/xtetris
-rwxr-xr-t 1 root root 27428 Nov 19 1996 /usr/X11R6/bin/xtetris
```

Absolute Permissions: Binary Masks

Instead of permission symbols, many users find it more convenient to use the absolute method. The *absolute method* changes all the permissions at once, instead of specifying one or the other. It uses a binary mask that references all the permissions in each category. The three categories, each with three permissions, conform to an octal binary format. Octal numbers have a base 8 structure. When translated into a binary number, each octal digit becomes three binary digits. A binary number is a set of 1 and 0 digits. Three octal digits in a number translate into three sets of three binary digits, which is nine altogether-and the exact number of permissions for a file.

You can use the octal digits as a mask to set the different file permissions. Each octal digit applies to one of the user categories. You can think of the digits matching up with the permission categories from left to right, beginning with the owner category. The first octal digit applies to the owner category, the second to the group, and the third to the others category.

The actual octal digit you choose determines the read, write, and execute permissions for each category. At this point, you need to know how octal digits translate into their binary equivalents. The following table shows how the different octal digits, 0-7, translate into their three-digit binary equivalents. You can think of the octal digit first being translated into its binary form, and then each of those three binary digits being used to set the read, write, and execute permissions. Each binary digit is then matched up with a corresponding permission, again moving from left to right. If a binary digit is 0, the permission is turned off. If the binary digit is 1, the permission is turned on. The first binary digit sets the read permission on or off, the second sets the write permission, and the third sets the execute permission. For example, an octal digit 6 translates into the binary digits 110. This would set the read and write permission on, but set the execute permission off.

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

When dealing with a binary mask, you need to specify three digits for all three categories, as well as their permissions. This makes a binary mask less versatile than the permission symbols. To set the owner execute permission on and the write permission off for the **mydata** file, as well as retain the read permission, you need to use the octal digit 5 (101). At the same time, you need to specify the digits for group and other users access. If these categories are to retain read access, you need the octal number 4 for each (100). This gives you three octal digits, 544, which translate into the binary digits 101 100 100.

```
$ chmod 544 mydata
```

One of the most common uses of the binary mask is to set the execute permission. As you learned in Chapter 11, you can create files that contain Linux commands; these files are called shell scripts. To have the commands in a shell script executed, you must first indicate the file is executable-that it contains commands the system can execute. You can do this in several ways, one of which is to set the executable permission on the shell script file. Suppose you just completed a shell script file and you need to give it executable permission to run it. You also want to retain read and write permission, but deny any access by the group or other users. The octal digit 7 (111) will set all three permissions, including execute (you can also add 4-read, 2-write, and 1-execute to get 7). Using 0 for the group and other users denies them access. This gives you the digits 700, which are equivalent to the binary digits 111 000 000. In the next example, the owner permission for the **myprog** file is set to include execute permission:

```
$ chmod 700 myprog
```

If you want others to be able to execute and read the file, but not change it, you can set the read and execute permissions and turn off the write permission with the digit 5 (101). In this case, you would use the octal digits 755, having the binary equivalent of 111 101 101.

```
$ chmod 755 myprog
```

A simple way to calculate the octal number makes use of the fact that any number used for permissions will be a combination derived from adding in decimal terms the numbers 4, 2, and 1. Use 4 for read permission, 2 for write, and 1 for execute. The read, write, execute permission is simply the addition of 4 + 2 + 1 to get 7. The read and execute permission adds 4 and 1, to get 5. You can use this method to calculate the octal number for each category. To get 755, you would add 4 + 2 + 1 for the user read, write, and execute permission, 4 + 1 for the group read and execute permission, and 4 + 1 again for the other read and execute permission.

For the ownership and sticky bit permissions, you add another octal number to the beginning of the octal digits. The octal digit for User ID permission is 4 (100); for Group ID, it is 2 (010); and for the sticky bit, it is 1 (001). The following example sets the User ID permission to the **pppd** program, along with read and execute permissions for the owner, group, and others:

```
# chmod 4555 /usr/sbin/pppd
```

The following example sets the sticky bit for the **xtetris** program:

```
# chmod 1755 /usr/X11R6/bin/xtetris
```

The next example would set both the sticky bit and the User ID permission on the **xman** program. The permission 5755 has the binary equivalent of 101 111 101 101.

```
# chmod 5755 /usr/X11R6/bin/xman
# ls -l /usr/X11R6/bin/xman
-rwsr-xr-t 1 root root 44364 Mar 26 04:28 /usr/X11R6/bin/xman
```

Directory Permissions

You can also set permissions on directories. The read permission set on a directory allows the list of files in a directory to be displayed. The execute permission enables a user to change to that directory. The write permission enables a user to create and remove his or her files in that directory. If you allow other users to have write permission on a directory, they can add their own files to it. When you create a directory, it is automatically given read, write, and execute permission for the owner. You may list the files in that directory, change to it, and create files in it.

Like files, directories have sets of permissions for the owner, the group, and all other users. Often, you may want to allow other users to change to and list the files in one of your directories, but not let them add their own files to it. In this case, you would set read and execute permissions on the directory, but not write permission. This would allow other users to change to the directory and list the files in it, but not to create new files or to copy any of their files into it. The next example sets read and execute permission for the group for the **thankyou** directory, but removes the write permission. Members of the group may enter the **thankyou** directory and list the files there, but they may not create new ones.

```
$ chmod g+rx-w letters/thankyou
```

Just as with files, you can also use octal digits to set a directory permission. To set the same permissions as in the previous example, you would use the octal digits 750, which have the binary equivalents of 111 101 000.

```
$ chmod 750 letters/thankyou
```

As you know, the **ls** command with the **-l** option will list all files in a directory. To list only the information about the directory itself, add a **d** modifier. In the next example, **ls -ld** displays information about the **thankyou** directory. Notice the first character in the permissions list is *d*, indicating it is a directory.

```
$ ls -ld thankyou
drwxr-x--- 2 chris 512 Feb 10 04:30 thankyou
```

If you have files you want other users to have access to, you not only need to set permissions for that file, you also must make sure the permissions are set for the directory in which the file is located. Another user, in order to access your file, must first access the file's directory. The same applies to parents of directories. Although a directory may give permission to others to access it, if its parent directory denies access, the directory cannot be reached. In this respect, you must pay close attention to your directory tree. To provide access to a directory, all other directories above it in the directory tree must also be accessible to other users.

Changing a File's Owner or Group: chown and chgrp

Although other users may be able to access a file, only the owner can change its permissions. If, however, you want to give some other user control over one of your file's permissions, you can change the owner of the file from yourself to the other user. The **chown** command transfers control over a file to another user. This command takes as its first argument the name of the other user. Following the user name, you can list the files you are giving up. In the next example, the user gives control of the **mydata** file to Robert:

```
$ chown robert mydata
$ ls -l mydata
-rw-r--r-- 1 robert weather 207 Feb 20 11:55 mydata
```

You can also, if you wish, change the group for a file, using the **chgrp** command. **chgrp** takes as its first argument the name of the new group for a file or files. Following the new group name, you list the files you want changed to that group. In the next example, the user changes the group name for **today** and **weekend** to the **forecast** group. The **ls -l** command then reflects the group change.

```
$ chgrp forecast today weekend
$ ls -l
-rw-r--r-- 1 chris weather 207 Feb 20 11:55 mydata
-rw-rw-r-- 1 chris forecast 568 Feb 14 10:30 today
-rw-rw-r-- 1 chris forecast 308 Feb 17 12:40 weekend
```

You can combine the **chgrp** operation in the **chown** command by attaching a group to the new owner with a colon.

Chapter 13: Shell Configuration

Overview

Four different major shells are commonly used on Linux systems: the Bourne Again shell (BASH), the Public Domain Korn shell (PDKSH), the TCSH shell, and the Z shell. The BASH shell is an advanced version of the Bourne shell, which includes most of the advanced features developed for the Korn shell and the C shell. TCSH is an enhanced version of the C shell, originally developed for BSD versions of UNIX. PDKSH is a subset of the UNIX Korn shell, whereas the Z shell is an enhanced version of the Korn shell. Although their UNIX counterparts differ greatly, the Linux shells share many of the same features. In UNIX, the Bourne shell lacks many capabilities found in the other UNIX shells. In Linux, however, the BASH shell incorporates all the advanced features of the Korn shell and C shell, as well as the TCSH shell. All four shells are available for your use, though the BASH shell is the default.

So far, all examples in this book have used the BASH shell. You log into your default shell, but you can change to another shell by entering its name. **tcsh** invokes the TCSH shell, **bash** the BASH shell, **ksh** the PDKSH shell, and **zsh** the Z shell. You can leave a shell with the CTRL-D or **exit** command. You only need one type of shell to do your work. This chapter describes common features of the BASH shell, such as history and aliases, as well as how to configure the shell to your own needs using shell variables and initialization files. The other shells share many of the same features, and use similar variables and initialization files.

Command and Filename Completion

The BASH command line has a built-in feature that performs command and filename completion. If you enter an incomplete pattern as a command or filename argument, you can then press the TAB key to activate the command and filename completion feature, which completes the pattern. If more than one command or file has the same prefix, the shell simply beeps and waits for you to add enough characters to select a unique command or filename. In the next example, the user issues a **cat** command with an incomplete filename. When you press the TAB key, the system searches for a match and, when it finds one, fills in the filename. The user can then press ENTER to execute the command.

```
$ cat pre tab
$ cat preface
```

The shell can also perform filename completion to list the partially matching files in your current directory. If you press ESC followed by a question mark, ESC-?, or press the TAB key again, the shell lists all filenames matching the incomplete pattern. In the next example, the ESC-? after the incomplete filename generates a list of possible filenames. The shell then redraws the command line, and you can type in the complete name of the file you want, or type in distinguishing characters and press the TAB key to have the filename completed.

```
$ ls
document docudrama
$ cat doc escape ?
document
docudrama
$ cat docudrama
```

Command Line Editing

The BASH shell has built-in command line editing capabilities that enable you to easily modify commands you have entered before executing them. If you make a spelling mistake when entering a command, rather than reentering the entire command, you can use the editing operations to correct the mistake before executing the command. This is most helpful for commands that use arguments with lengthy pathnames. The command line editing operations are implemented by Readline, which uses a subset of the Emacs editing commands (see Table 13-1). You can use CTRL-F or the RIGHT ARROW key to move forward a character, and the CTRL-B or the LEFT ARROW key to move back a character. CTRL-D or DEL deletes the character the cursor is on. To add text, you move the cursor to where you want to insert text and type in the new characters. At any time, you can press ENTER to execute the command.

Table 13-1: Command Line Editing, History Commands, and History Event References

Command Line Editing	**Description**
CTRL-B or LEFT ARROW	Moves left one character (backward to the previous character)
CTRL-F or RIGHT ARROW	Moves right one character (forward to the next character)
CTRL-A	Moves to the beginning of a line
CTRL-E	Moves to the end of a line
ESC-F	Moves forward one word
ESC-B	Moves backward one word
DEL	Deletes the character the cursor is on
BACKSPACE or CTRL-H	Deletes the character before the cursor

Table 13-1: Command Line Editing, History Commands, and History Event References

Command Line Editing	**Description**
CTRL-D	Deletes the character the cursor is on
CTRL-K	Removes (kills) the remainder of a line
History Commands	
CTRL-N or DOWN ARROW	Moves down to the next event in the history list
CTRL-P or UP ARROW	Moves up to the previous event in the history list
ESC-	Moves to the beginning of the history event list
ESC-	Moves to the end of the history event list
ESC-TAB	History of event matching and completion
fc *event-reference*	Edits an event with the standard editor and then executes it **options** **-l** List recent history events; same as **history** command **-e** *editor event-reference* Invokes a specified editor to edit a specific event
History Event References	
!*event num*	References an event with an event number
!*characters*	References an event with beginning characters
!?*pattern***?**	References an event with a pattern in the event
!-*event num*	References an event with an offset from the first event
!*num-num*	References a range of events

Note As described in the next section, you can also use the command line editing operations to modify history events-previous commands you have entered.

History

In the BASH shell, the *history utility* keeps a record of the most recent commands you have executed. The commands are numbered starting at 1, and a limit exists to the number of commands remembered-the default is 500. The history utility is a kind of short-term memory, keeping track of the most recent commands you have executed. To see the set of your most recent commands, type **history** on the command line and press ENTER. A list of your most recent commands is then displayed, preceded by a number.

```
$ history
1 cp mydata today
2 vi mydata
3 mv mydata reports
4 cd reports
5 ls
```

Each of these commands is technically referred to as an "event." An *event* describes an action that has been taken-a command that has been executed. The events are numbered according to their sequence of execution. The most recent event has the highest number. Each of these events can be identified by its number or beginning characters in the command.

The history utility enables you to reference a former event, placing it on your command line and enabling you to execute it. The easiest way to do this is to use the UP ARROW and DOWN ARROW keys to place history events on your command line, one at a time. You needn't display the list first with **history**. Pressing the UP ARROW key once places the last history event on your command line. Pressing it again places the next history event on your command line. Pressing the DOWN ARROW key places the previous event on the command line.

The BASH shell also has a history event completion operation invoked by the ESC-TAB command. Much like standard command line completion, you enter part of the history event you want. Then you press ESC, followed by TAB. The event that matches the text you have entered is then located and used to complete your command line entry.

Note If more than one history event matches what you have entered, you will hear a beep, and you can then enter more characters to help uniquely identify the event you want.

You can edit the event displayed on your command line using the command line editing operations. The LEFT ARROW and RIGHT ARROW keys move you along the command line. You can insert text wherever you stop your cursor. With BACKSPACE and DEL, you can delete characters. Once the event is displayed on your command line, you can press ENTER to execute it.

You can also reference and execute history events using the **!** history command. The **!** is followed by a reference that identifies the command. The reference can be either the number of the event or a beginning set of characters in the event. In the next example, the third command in the history list is referenced first by number and then by the beginning characters:

```
$ !3
mv mydata reports
$ !mv
mv mydata reports
```

You can also reference an event using an offset from the end of the list. A negative number will offset from the end of the list to that event, thereby referencing it. In the next example, the fourth command, **cd mydata**, is referenced using a negative offset, and then executed. Remember that you are offsetting from the end of the list-in this case, event 5, up toward the beginning of the list, event 1. An offset of 4 beginning from event 5 places you at event 2.

```
$ !-4
vi mydata
```

If no event reference is used, then the last event is assumed. In the next example, the command **!** by itself executes the last command the user executed-in this case, **ls**:

```
$ !
ls
mydata today reports
```

History Event Editing

You can also edit any event in the history list before you execute it. In the BASH shell, you can do this two ways. You can use the command line editor capability to reference and edit any event in the history list. You can also use a history **fc** command option to reference an event and edit it with the full Vi editor. Each approach involves two different editing capabilities. The first is limited to the commands in the command line editor, which edits only a single line with a subset of Emacs commands. At the same time, however, it enables you to reference events easily in the history list. The second approach invokes the standard Vi editor with all its features, but only for a specified history event.

With the command line editor, not only can you edit the current command, but you can also move to a previous event in the history list to edit and execute it. The CTRL-P command then moves you up to the prior event in the list. The CTRL-N command moves you down the list. The ESC-< command moves you to the top of the list, and the ESC-> command moves you to the bottom. You can even use a pattern to search for a given event. The slash followed by a pattern searches backward in the list, and the question mark followed by a pattern searches forward in the list. The **n** command repeats the search.

Once you locate the event you want to edit, you use the Emacs command line editing commands to edit the line. CTRL-D deletes a character. CTRL-F or the RIGHT ARROW moves you forward a character, and CTRL-B or the LEFT ARROW moves you back a character. To add text, you position your cursor and type in the characters you want. Table 13-1 lists the different commands for referencing the history list.

If you want to edit an event using a standard editor instead, you need to reference the event using the **fc** command and a specific event reference, such as an event number. The editor used is the one specified by the shell in the **EDITOR** variable. This serves as the default editor for the **fc** command. You can assign to the **EDITOR** variable a different editor if you wish, such as **emacs** instead of **vi**. The next example will edit the fourth event, **cd reports**, with the standard editor and then execute the edited event:

```
$ fc 4
```

You can select more than one command at a time to be edited and executed by referencing a range of commands. You select a range of commands by indicating an identifier for the first command followed by an identifier for the last command in the range. An identifier can be the command number or the beginning characters in the command. In the next example, the range of commands 2-4 are edited and executed, first using event numbers and then using beginning characters in those events:

```
$ fc 2 4
$ fc vi c
```

fc uses the default editor specified in the **FCEDIT** special variable. Usually, this is the Vi editor. If you want to use the Emacs editor instead, you use the **-e** option and the term **emacs** when you invoke **fc**. The next example will edit the fourth event, **cd reports**, with the Emacs editor and then execute the edited event:

```
$ fc -e emacs 4
```

Configuring History: HISTFILE and HISTSAVE

The number of events saved by your system is kept in a special system variable called **HISTSIZE**. By default, this is usually set to 500. You can change this to another number by simply assigning a new value to **HISTSIZE**. In the next example, the user changes the number of history events saved to 10 by resetting the **HISTSIZE** variable:

```
$ HISTSIZE=10
```

The actual history events are saved in a file whose name is held in a special variable called **HISTFILE**. By default, this file is the **.bash_history** file. You can change the file in which history events are saved, however, by assigning its name to the **HISTFILE** variable. In the next example, the value of **HISTFILE** is displayed. Then a new filename is assigned to it, **newhist**. History events are then saved in the **newhist** file.

```
$ echo $HISTFILE
.bash_history
$ HISTFILE="newhist"
$ echo $HISTFILE
newhist
```

Aliases

You use the **alias** command to create another name for a command. The **alias** command operates like a macro that expands to the command it represents. The alias does not literally replace the name of the command; it simply gives another name to that command. An **alias** command begins with the keyword **alias** and the new name for the command, followed by an equal sign and the command the alias will reference.

Note No spaces can be around the equal sign used in the **alias** command.

In the next example, **list** becomes another name for the **ls** command:

```
$ alias list=ls
$ ls
mydata today
$ list
mydata today
$
```

You can also use an alias to substitute for a command and its option, but you need to enclose both the command and the option within single quotes. Any command you alias that contains spaces must be enclosed in single quotes. In the next example, the alias **lss** references the **ls** command with its **-s** option, and the alias **lsa** references the **ls** command with the **-F** option. **ls** with the **-s** option lists files and their sizes in blocks, and the **ls** with the **-F** option places a slash after directory names. Notice single quotes enclose the command and its option.

```
$ alias lss='ls -s'
$ lss
mydata 14   today  6   reports  1
$ alias lsa='ls -F'
$ lsa
mydata today reports/
```

```
$
```

You may often use an alias to include a command name with an argument. If you execute a command that has an argument with a complex combination of special characters on a regular basis, you may want to alias it. For example, suppose you often list just your source code and object code files-those files ending in either a **.c** or **.o**. You would need to use as an argument for **ls** a combination of special characters such as ***.[co]**. Instead, you could alias **ls** with the ***.[co]** argument, giving it a simple name. In the next example, the user creates an alias called **lsc** for the command **ls*.[co]**:

```
$ alias lsc='ls *.[co]'
$ lsc
main.c main.o lib.c lib.o
```

You can also use the name of a command as an alias. This can be helpful in cases where you should only use a command with a specific option. In the case of the **rm**, **cp**, and **mv** commands, the **-i** option should always be used to ensure an existing file is not overwritten. Instead of constantly being careful to use the **-i** option each time you use one of these commands, the command name can be aliased to include the option. In the next example, the **rm**, **cp**, and **mv** commands have been aliased to include the **-i** option:

```
$ alias rm='rm -i'
$ alias mv='mv -i'
$ alias cp='cp -i'
```

The **alias** command by itself provides a list of all aliases that have been defined, showing the commands they represent. You can remove an alias by using the **unalias** command. In the next example, the user lists the current aliases and then removes the **lsa** alias:

```
$ alias
lsa=ls -F
list=ls
rm=rm -i
$ unalias lsa
```

Controlling Shell Operations

The BASH shell has several features that enable you to control the way different shell operations work. For example, setting the **noclobber** feature prevents redirection from overwriting files. You can turn these features on and off like a toggle, using the **set** command. The **set** command takes two arguments: an option specifying on or off and the name of the feature. To set a feature on, you use the **-o** option, and to set it off, you use the **+o** option. Here is the basic form:

```
$ set -o feature        turn the feature on
$ set +o feature        turn the feature off
```

Three of the most common features are described here: **ignoreeof**, **noclobber**, and **noglob**. Table 13-2 lists these different features, as well as the **set** command. Setting **ignoreeof** enables a feature that prevents you from logging out of the user shell with a CTRL-D. CTRL-D is used not only to log out of the user shell, but also to end user input entered directly into the standard input. CTRL-D is used often for the Mail program or for utilities such as cat. You could easily enter an extra CTRL-D in such circumstances and accidentally log yourself out.

The **ignoreeof** feature prevents such accidental logouts. In the next example, the **ignoreeof** feature is turned on using the **set** command with the **-o** option. The user can now only log out by entering the **logout** command.

```
$ set -o ignoreeof
$ ctrl-d
Use exit to logout
$
```

Setting **noclobber** enables a feature that safeguards existing files from redirected output. With the **noclobber** feature, if you redirect output to a file that already exists, the file will not be overwritten with the standard output. The original file is preserved. Situations may occur in which you use, as the name for a file to hold the redirected output, a name you have already given to an existing file. The **noclobber** feature prevents you from accidentally overwriting your original file. In the next example, the user sets the **noclobber** feature on and then tries to overwrite an existing file, **myfile**, using redirection. The system returns an error message.

```
$ set -o noclobber
$ cat preface > myfile
myfile: file exists
$
```

At times, you may want to overwrite a file with redirected output. In this case, you can place an exclamation point after the redirection operator. This will override the **noclobber** feature, replacing the contents of the file with the standard output.

```
$ cat preface >! myfile
```

Setting **noglob** enables a feature that disables special characters in the user shell. The characters *, ?, [], and ~ will no longer expand to matched filenames. This feature is helpful if you have special characters as part of the name of a file. In the next example, the user needs to reference a file that ends with the **?** character, **answers?**. First, the user turns off special characters using the **noglob** feature. Now the question mark on the command line is taken as part of the filename, not as a special character, and the user can reference the **answers?** file.

```
$ set -o noglob
$ ls answers?
answers?
```

Environment Variables and Subshells: export

When you log into your account, Linux generates your user shell. Within this shell, you can issue commands and declare variables. You can also create and execute shell scripts. When you execute a shell script, however, the system generates a subshell. You then have two shells, the one you logged into and the one generated for the script. Within the script shell, you could execute another shell script, which would have its own shell. When a script has finished execution, its shell terminates and you return to the shell from which it was executed. In this sense, you can have many shells, each nested within the other. Variables you define within a shell are local to it. If you define a variable in a shell script, then, when the script is run, the variable is defined with that script's shell and is local to it. No other shell can reference that variable. In a sense, the variable is hidden within its shell.

You can define environment variables in all types of shells including the BASH, the Z shell, and the TCSH shell. The strategy used to implement environment variables in the BASH shell, however, is different from that of the TCSH shell. In the BASH shell, environment variables are exported. That is to say, a copy of an environment variable is made in each subshell. For example, if the **EDITOR** variable is exported, a copy is automatically defined in each subshell for you. In the TCSH shell, on the other hand, an environment variable is defined only once and can be directly referenced by any subshell.

In the BASH shell, an environment variable can be thought of as a regular variable with added capabilities. To make an environment variable, you apply the **export** command to a variable you have already defined. The **export** command instructs the system to define a copy of that variable for each new shell generated. Each new shell will have its own copy of the environment variable. This process is called *exporting* variables. Thinking of exported environment variables as global variables is a mistake. A new shell can never reference a variable outside of itself. Instead, a copy of the variable with its value is generated for the new shell.

Note You can think of exported variables as exporting their values to a shell, not to themselves. For those familiar with programming structures, exported variables can be thought of as a form of "call by value."

Configuring Your Shell with Special Shell Variables

When you log into your account, the system generates a shell for you. This shell is referred to as either your login shell or your user shell. When you execute scripts, you are generating subshells of your user shell. You can define variables within your user shell, and you can also define environment variables that can be referenced by any subshells you generate. Linux sets up special shell variables you can use to configure your user shell. Many of these special shell variables are defined by the system when you log in, but you define others yourself. See Table 13-2 for a list of the commonly used ones.

Table 13-2: BASH Shell Special Variables and Features

BASH Shell Special Variables	Description
HOME	Pathname for user's home directory
LOGNAME	Login name
USER	Login name
SHELL	Pathname of program for type of shell you are using
BASH_ENV	Holds name of BASH initialization script executed whenever a BASH shell script is run or BASH shell entered. Usually **$HOME/.bashrc**
PATH	List of pathnames for directories searched for executable commands
PS1	Primary shell prompt
PS2	Secondary shell prompt
IFS	Interfield delimiter symbol

Table 13-2: BASH Shell Special Variables and Features

BASH Shell Special Variables	**Description**
MAIL	Name of mail file checked by mail utility for received messages
MAILCHECK	Interval for checking for received mail
MAILPATH	List of mail files to be check by mail for received messages
TERM	Terminal name
CDPATH	Pathnames for directories searched by **cd** command for subdirectories
EXINIT	Initialization commands for Ex/Vi editor
BASH Shell Features	
$ set **-+o** *feature*	Bash shell features are turned on and off with the **set** command; **-o** sets a feature on and **+o** turns it off: **$ set -o noclobber** *set noclobber on* **$ set +o noclobber** *set noclobber off*
ignoreeof	Disabled CTRL-D logout
noclobber	Does not overwrite files through redirection
noglob	Disables special characters used for filename expansion: *****, **?**, **~**, and **[]**

A reserved set of keywords is used for the names of these special variables. You should not use these keywords as the names of any of your own variable names. The special shell variables are all specified in uppercase letters, making them easy to identify. Shell feature variables are in lowercase letters. For example, the keyword **HOME** is used by the system to define the **HOME** variable. **HOME** is a special environment variable that holds the pathname of the user's home directory. On the other hand, the keyword **noclobber**, covered earlier in the chapter, is used to set the **noclobber** feature on or off.

Common Special Variables

Many of the special variables automatically defined and assigned initial values by the system when you log in can be changed, if you wish. Some special variables exist whose values should not be changed, however. For example, the **HOME** variable holds the pathname for your home directory. Commands, such as **cd,** reference the pathname in the **HOME** special variable to locate your home directory. Some of the more common of these special variables are described in this section. Other special variables are defined by the system and given an initial value that you are free to change. To do this, you redefine them and assign a new value. For example, the **PATH** variable is defined by the system and given an initial value; it contains the pathnames of directories where commands are located. Whenever you execute a command, the shell searches for it in these directories. You can add a new directory to be searched by redefining the **PATH** variable yourself, so it will include the new directory's pathname. Still other special variables exist that the system does not define. These are usually optional features, such as the **EXINIT** variable that enables you to set options for the Vi editor. Each time you log in, you must define and assign a value to such variables.

Note You can obtain a listing of the currently defined special variables using the **env** command. The **env** command operates like the **set** command, but it only lists special variables.

You can automatically define special variables using special shell scripts called initialization files. An *initialization file* is a specially named shell script executed whenever you enter a certain shell. You can edit the initialization file and place in it definitions and assignments for special variables. When you enter the shell, the initialization file will execute these definitions and assignments, effectively initializing special variables with your own values. For example, the BASH shell's **.bash_profile** file is an initialization file executed every time you log in. It contains definitions and assignments of special variables. However, the **.bash_profile** file is basically only a shell script, which you can edit with any text editor such as the Vi editor, changing, if you wish, the values assigned to special variables.

In the BASH shell, all the special variables are designed to be environment variables. When you define or redefine a special variable, you also need to export it to make it an environment variable. This means any change you make to a special variable must be accompanied by an **export** command. You shall see that at the end of the login initialization file, **.bash_profile**, there is usually an **export** command for all the special variables defined in it.

The **HOME** variable contains the pathname of your home directory. Your home directory is determined by the system administrator when your account is created. The pathname for your home directory is automatically read into your **HOME** variable when you log in. In the next example, the **echo** command displays the contents of the **HOME** variable:

```
$ echo $HOME
/home/chris
```

The **HOME** variable is often used when you need to specify the absolute pathname of your home directory. In the next example, the absolute pathname of **reports** is specified using **HOME** for the home directory's path:

```
$ ls $HOME/reports
```

Some of the more common special variables are **SHELL**, **PATH**, **PS1**, **PS2**, and **MAIL**. The **SHELL** variable holds the pathname of the program for the type of shell you log into. The **PATH** variable lists the different directories to be searched for a Linux command. The **PS1** and **PS2** variables hold the prompt symbols. The **MAIL** variable holds the pathname of your mailbox file. You can modify the values for any of them to customize your shell.

The **PATH** variable contains a series of directory paths separated by colons. Each time a command is executed, the paths listed in the **PATH** variable are searched one by one for that command. For example, the **cp** command resides on the system in the directory **/usr/bin**. This directory path is one of the directories listed in the **PATH** variable. Each time you execute the **cp** command, this path is searched and the **cp** command located. The system defines and assigns **PATH** an initial set of pathnames. In Linux, the initial pathnames are **/usr/bin** and **usr/sbin**.

The shell can execute any executable file, including programs and scripts you have created. For this reason, the **PATH** variable can also reference your working directory; so if you want to execute one of your own scripts or programs in your working directory, the shell can locate it. No spaces can be between the pathnames in the string. A colon with no pathname specified references your working directory. Usually, a single colon is placed at the end of the pathnames as an empty entry specifying your working directory. For example, the pathname **/usr/bin:/usr/sbin:** references three directories: **/usr/bin**, **/usr/sbin**, and your current working directory.

```
$ echo $PATH
/usr/bin:/usr/sbin:
```

You can add any new directory path you want to the **PATH** variable. This can be useful if you have created several of your own Linux commands using shell scripts. You could place these new shell script commands in a directory you created and then add that directory to the **PATH** list. Then, no matter what directory you are in, you can execute one of your shell scripts. The **PATH** variable will contain the directory for that script, so that directory will be searched each time you issue a command.

You add a directory to the **PATH** variable with a variable assignment. You can execute this assignment directly in your shell. In the next example, the user **chris** adds a new directory, called **mybin,** to the **PATH**. Although you could carefully type in the complete pathnames listed in **PATH** for the assignment, you can also use an evaluation of **PATH**, **$PATH**, in their place. In this example, an evaluation of **HOME** is also used to designate the user's **home** directory in the new directory's pathname. Notice the empty entry between two colons, which specifies the working directory.

```
$ PATH=$PATH:$HOME/mybin:
$ export PATH
$ echo $PATH
/usr/bin:/usr/sbin::/home/chris/mybin
```

If you add a directory to **PATH** yourself while you are logged in, the directory would be added only for the duration of your login session. When you log back in, the login initialization file, **.bash_profile**, would again initialize your **PATH** with its original set of directories. The **.bash_profile** file is described in detail a bit later in this chapter. To add a new directory to your **PATH** permanently, you need to edit your **.bash_profile** file and find the assignment for the **PATH** variable. Then, you simply insert the directory, preceded by a colon, into the set of pathnames assigned to **PATH**.

The **BASH_ENV** variable holds the name of the BASH shell initialization file to be executed whenever a BASH shell is generated. For example, when a BASH shell script is executed, the **BASH_ENV** variable is checked and the name of the script that it holds is executed before the shell script. On Red Hat Linux, the **BASH_ENV** variable holds **$HOME/.bashrc**. This is the **.bashrc** file in the user's home directory. The **.bashrc** file is discussed later in this chapter. You could specify a different file if you wish, using that instead of the **.bashrc** file for BASH shell scripts.

The **PS1** and **PS2** variables contain the primary and secondary prompt symbols, respectively. The primary prompt symbol for the BASH shell is a dollar sign, **$**. You can change the prompt symbol by assigning a new set of characters to the **PS1** variable. In the next example, the shell prompt is changed to the **->** symbol:

```
$ PS1="->"
-> export PS1
```

```
->
```

You can change the prompt to be any set of characters, including a string, as shown in the next example:

```
$ PS1="Please enter a command: "
Please enter a command: export PS1
Please enter a command: ls
mydata /reports
Please enter a command:
```

The **PS2** variable holds the secondary prompt symbol, which is used for commands that take several lines to complete. The default secondary prompt is >. The added command lines begin with the secondary prompt instead of the primary prompt. You can change the secondary prompt just as easily as the primary prompt, as shown here:

```
$ PS2="@"
```

Like the TCSH shell, the BASH shell provides you with a predefined set of codes you can use to configure your prompt. With them you can make the time, your user name, or your directory pathname a part of your prompt. You can even have your prompt display the history event number of the current command you are about to enter. Each code is preceded by a \ symbol. **\w** represents the current working directory, **\t** the time, and **\u** your user name. **\!** will display the next history event number. In the next example, the user adds the current working directory to the prompt:

```
$ PS1="\w $"
/home/dylan $
```

The codes must be included within a quoted string. If no quotes exist, the code characters are not evaluated and are themselves used as the prompt. **PS1=\w** sets the prompt to the characters **\w**, not the working directory. The next example incorporates both the time and the history event number with a new prompt:

```
$ PS1="\t \! ->"
```

The following table lists the codes for configuring your prompt:

Prompt Codes	Description
\!	Current history number.
\$	Use $ as prompt for all users except the root user, which has the # as its prompt.
\d	Current date.
\s	Shell currently active.
\t	Time of day.
\u	User name.
\w	Current working directory.

If **CDPATH** is undefined, then when the **cd** command is given a directory name as its argument, it searches only the current working directory for that name. If **CDPATH** is defined, however, **cd** also searches the directories listed in **CDPATH** for that directory name. If the directory name is found, **cd** changes to that directory. This is helpful if you are working on a project in which you constantly must change to directories in another part of the file system. To change to a directory that has a pathname very different from the one you are in, you would need to know the full pathname of that directory. Instead, you could simply place the pathname of that directory's parent in **CDPATH**. Then, **cd** automatically searches the parent directory, finding the name of the directory you want.

Note Notice that you assign to **CDPATH** the pathname of the parent of the directory you want to change to, not the pathname of the directory itself.

Using the **HOME** variable to specify the user's home directory part of the path in any new pathname added to **CDPATH** is advisable. This is because the pathname for your home directory could possibly be changed by the system administrator during a reorganization of the file system. **HOME** will always hold the current pathname of the user's home directory. In the next example, the pathname **/home/chris/letters** is specified with **$HOME/letters**:

```
$ CDPATH=$CDPATH:$HOME/letters
$ export CDPATH
$ echo $CDPATH
:/home/chris/letters
```

Several shell special variables are used to set values used by network applications, such as Web browsers or newsreaders. **NNTPSERVER** is used to set the value of a remote news server accessible on your network. If you are using an ISP, the ISP usually provides a news server you can access with your newsreader applications. However, you first have to provide your newsreaders with the Internet address of the news server. This is the role of the **NNTPSERVER**. News servers on the Internet usually use the NNTP protocol. **NNTPSERVER** should hold the address of such a news server. For many ISPs, the news server address is a domain name that begins with **nntp**. The following example assigns the news server address **nntp.myservice.com** to the **NNTPSERVER** special variables. Newsreader applications automatically obtain the news server address from **NNTPSERVER**. Usually, this assignment is placed in the shell initialization file, **.bash_profile**, so it is automatically set each time a user logs in.

```
NNTPSERVER=nntp.myservice.com
export NNTPSERVER
```

Other special variables are used for specific applications. The **KDEDIR** variable holds the pathname for the KDE Desktop program files. This is usually **/opt/kde** but, at the time of installation, you can choose to install KDE in a different directory and then change the value of **KDEDIR** accordingly.

```
export KDEDIR=/opt/kde
```

Configuring Your Login Shell: .bash_profile

The **.bash_profile** file is the BASH shell's login initialization file, which can also be named **.profile**. It is a script file that is automatically executed whenever a user logs in. The file contains shell commands that define special environment variables used to manage your shell. They may be either redefinitions of system-defined special variables or definitions of user-defined special variables. For example, when you log in, your user shell needs to know what directories hold Linux commands. It will reference the **PATH** variable to find the pathnames for these directories. However, first, the **PATH** variable must be assigned those pathnames. In the **.bash_profile** file, an assignment operation does just this. Because it is in the **.bash_profile** file, the assignment is executed automatically when the user logs in.

Special variables also need to be exported, using the **export** command, to make them accessible to any subshells you may enter. You can export several variables in one **export** command by listing them as arguments. Usually, at the end of the **.bash_profile** file is an **export** command with a list of all the variables defined in the file. If a variable is missing from this list, you may be unable to access it. Notice the **export** command at the end of the **.profile** file in the example described next. You can also combine the assignment and **export** command into one operation as show here for **NNTPSERVER**:

```
export NNTPSERVER=nntp.myservice.com
```

A copy of the standard **.bash_profile** file provided for you when your account is created is listed in the next example. Notice how **PATH** is assigned, as is the value of **$HOME**. Both **PATH** and **HOME** are system special variables the system has already defined. **PATH** holds the pathnames of directories searched for any command you enter, and **HOME** holds the pathname of your home directory. The assignment **PATH=$PATH:$HOME/ bin** has the effect of redefining **PATH** to include your **bin** directory within your home directory. So, your **bin** directory will also be searched for any commands, including ones you create yourself, such as scripts or programs. Notice **PATH** is then exported, so it can be accessed by any subshells. Should you want to have your home directory searched also, you can use any text editor to modify this line in your **.bash_profile** file to **PATH=$PATH:$HOME/bin:$HOME**, adding **:$HOME** at the end. In fact, you can change this entry to add as many directories as you want searched.

.bash_profile

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin
BASH_ENV=$HOME/.bashrc
USERNAME=""

export USERNAME BASH_ENV PATH
```

Your Linux system also has its own profile file that it executes whenever any user logs in. This system initialization file is simply called **profile** and is found in the **/etc** directory,

/etc/profile. This file contains special variable definitions the system needs to provide for each user. A copy of the system's **.profile** file follows. Notice how **PATH** is redefined to include the **/usr/X11R6/bin** directory. This is the directory that holds the X Windows commands you execute when using the desktop. Also, includes the pathname for the KDE Desktop programs, **/opt/kde/bin**. **HISTSIZE** is also redefined to include a larger number of history events. An entry has been added here for the **NNTPSERVER** variable. Normally, a news server address is a value that needs to be set for all users. Such assignments should be made in the system's **/etc/profile** file by the system administrator, rather than in each individual user's own **.bash_profile** file. The **/etc/profile** file also executes any scripts in the directory **/etc/profile.d**. This design allows for a more modular structure. Rather than make entries by editing the **/etc/profile** file, you can just add a script to **profile.d** directory. The scripts for the BASH shell have the extension **.sh**. For example, the **kde.sh** script in the **profile.d** directory checks for a definition of the **KDEDIR** variable and makes one if none is in effect.

/etc/profile

```
# /etc/profile

# System wide environment and startup programs
# Functions and aliases go in /etc/bashrc
if [ `id -u` = 0 ] && ! echo $PATH | /bin/grep -q "/sbin" ; then
    PATH=/sbin:$PATH
fi
if [ `id -u` = 0 ] && ! echo $PATH | /bin/grep -q "/usr/local/sbin" ; then
    PATH=/usr/local/sbin:$PATH
fi
if ! echo $PATH | /bin/grep -q "/usr/X11R6/bin" ; then
  PATH="$PATH:/usr/X11R6/bin"

fi PS1="[\u@\h \W]\\$ "

USER=`id -un`
LOGNAME=$USER
MAIL="/var/spool/mail/$USER"

HOSTNAME=`/bin/hostname`
HISTSIZE=1000
NNTPSERVER=nntp.myservice.com

if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ]; then
    INPUTRC=/etc/inputrc
fi

export PATH PS1 HOSTNAME HISTSIZE MAIL NNTPSERVER

for i in /etc/profile.d/*.sh ; do
   if [ -x $i ]; then
      .$i
   fi
done

unset i
```

Your **.bash_profile** initialization file is a text file that can be edited by a text editor, like any other text file. You can easily add new directories to your **PATH** by editing **.bash_profile** and using editing commands to insert a new directory pathname in the list of directory pathnames assigned to the **PATH** variable. You can even add new variable definitions. If you do so, however, be sure to include the new variable's name in the **export** command's argument list. For example, if your **.bash_profile** file does not have any definition of the **EXINIT** variable, you can edit the file and add a new line that assigns a value to **EXINIT**. The definition **EXINIT='set nu ai'** will configure the Vi editor with line numbering and indentation. You then need to add **EXINIT** to the **export** command's argument list. When the **.bash_profile** file executes again, the **EXINIT** variable will be set to the command **set nu ai**. When the Vi editor is invoked, the command in the **EXINIT** variable will be executed, setting the line number and auto-indent options automatically.

In the following example, the user's **.bash_profile** has been modified to include definitions of **EXINIT** and redefinitions of **PATH**, **CDPATH**, **PS1**, and **HISTSIZE**. The **PATH** variable has **$HOME**: added to its value. **$HOME** is a variable that evaluates to the user's home directory, and the ending colon specifies the current working directory, enabling you to execute commands that may be located in either the home directory or the working directory. The redefinition of **HISTSIZE** reduces the number of history events saved, from 1,000 defined in the system's **.profile** file to 30. The redefinition of the **PS1** special variable changes the prompt to include the pathname of the current working directory. Any changes you make to special variables within your **.bash_profile** file override those made earlier by the system's **.profile** file. All these special variables are then exported with the **export** command.

.bash_profile

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ];
 then
    .~/.bashrc
fi
# User-specific environment and startup programs
PATH=/usr/local/sbin:/usr/sbin:/sbin:$PATH:$HOME/bin:$HOME:
BASH_ENV=$HOME/.bashrc
USERNAME=""
CDPATH=$CDPATH:$HOME/bin:$HOME
HISTSIZE=30
NNTPSERVER=nntp.myserver.com
EXINIT='set nu ai'
PS1="\w \$"
export USERNAME BASH_ENV PATH CDPATH HISTSIZE EXINIT PS1
```

Although **.bash_profile** is executed each time you log in, it is not automatically reexecuted after you make changes to it. The **.bash_profile** file is an initialization file that is *only* executed whenever you log in. If you want to take advantage of any changes you make to it without having to log out and log in again, you can reexecute **.bash_profile** with the dot (**.**) command. The **.bash_profile** file is a shell script and, like any shell script, can be executed with the **.** command.

```
$ . .bash_profile
```

Alternatively, you can use the **source** command to execute the **.bash_profile** initialization file, or any initialization file such as **.login** used in the TCSH shell, or **.bashrc**.

```
$ source .bash_profile
```

Configuring the BASH Shell: .bashrc

The **.bashrc** file is a configuration file executed each time you enter the BASH shell or generate any subshells. If the BASH shell is your login shell, **.bashrc** is executed along with your **.bash_login** file when you log in. If you enter the BASH shell from another shell, the **.bashrc** file is automatically executed, and the variable and alias definitions it contains will be defined. If you enter a different type of shell, the configuration file for that shell will be executed instead. For example, if you were to enter the TCSH shell with the **tcsh** command, the **.tcshrc** configuration file is executed instead of **.bashrc**.

The **.bashrc** shell configuration file is actually executed each time you generate a BASH shell, such as when you run a shell script. In other words, each time a subshell is created, the **.bashrc** file is executed. This has the effect of exporting any local variables or aliases you have defined in the **.bashrc** shell initialization file. The **.bashrc** file usually contains the definition of aliases and any feature variables used to turn on shell features. Aliases and feature variables are locally defined within the shell. But the **.bashrc** file defines them in every shell. For this reason, the **.bashrc** file usually holds such aliases as those defined for the **rm**, **cp**, and **mv** commands. The next example is a **.bashrc** file with many of the standard definitions:

.bashrc

```
# Source global definitions

if [ -f /etc/bashrc ];
 then
    .  /etc/bashrc
fi
set  -o ignoreeof
set  -o noclobber
alias rm 'rm -i'
alias mv 'mv -i'
alias cp 'cp -i'
```

Linux systems usually contain a system **.bashrc** file executed for all users. This may contain certain global aliases and features needed by all users whenever they enter a BASH shell. This is located in the **/etc** directory, **/etc/.bashrc**. A user's own **.bashrc** file, located in the **home** directory, contains commands to execute this system **.bashrc** file. The **./ etc/bashrc** command in the previous example of **.bashrc** does just that. You can add any commands or definitions of your own to your **.bashrc** file. If you have made changes to **.bashrc** and you want them to take effect during your current login session, you need to reexecute the file with either the **.** or the **source** command.

```
$ . .bashrc
```

The BASH Shell Logout File: .bash_logout

The **.bash_logout** file is also a configuration file, which is executed when the user logs out. It is designed to perform any operations you want done whenever you log out. Instead of variable definitions, the **.bash_logout** file usually contains shell commands that form a kind of shutdown procedure-actions you always want taken before you log out. One common logout command is to clear the screen and then issue a farewell message.

As with **.bash_profile**, you can add your own shell commands to **.bash_logout**. In fact, the **.bash_logout** file is not automatically set up for you when your account is first created. You need to create it yourself, using the Vi or Emacs editor. You could then add a farewell message or other operations. In the next example, the user has a **clear** and an **echo** command in the **.bash_logout** file. When the user logs out, the **clear** command clears the screen, and then the **echo** command displays the message "Good-bye for now."

.bash_logout

```
 # ~/.bash_logout

clear
echo "Good-bye for now"
```

Other Initialization and Configuration Files

Each type of shell has is its own set of initialization and configuration files. The TCSH shell used **.login**, **.tcshrc**, **.logout** files in place of **.bash_profile**, **.bashrc**, and **.bash_logout**. The Z shell has several initialization files: **.zshenv**, **.zlogin**, **.zprofile**, **.zschrc**, and **.zlogout**. See Table 13-3 for a listing. Check the Man pages for each shell to see how they are usually configured. When you install a shell, default versions of these files are automatically placed in the users' home directories. Except for the TCSH shell, all shells use much the same syntax for variable definitions and assigning values (TCSH uses a slightly different syntax, described in its Man pages).

Table 13-3: Shell Configuration Files

BASH Shell	**Function**
.bash_profile	Login initialization file
.bashrc	BASH shell configuration file
.bash_logout	Logout name
TCSH Shell	
.login	Login initialization file
.tcshrc	TCSH shell configuration file
.logout	Logout file
Z-shell	
.zshenv	Shell login file (first read)

Table 13-3: Shell Configuration Files

BASH Shell	**Function**
.zprofile	Login initialization file
.zlogin	Shell login file
.zshrc	Z shell shell configuration file
.zlogout	Logout file
PDKSH Shell	
.profile	Login initialization file
.kshrc	PDKSH shell configuration file

Configuration Directories and Files

Applications often install configuration files in a user's home directory that contain specific configuration information, which tailors the application to the needs of that particular user. This may take the form of a single configuration file that begins with a period, or a directory that contains several configuration files. The directory name will also begin with a period. For example, Netscape installs a directory called **.netscape** in the user's home directory that contains configuration files. On the other hand, the Mail application uses a single file called **.mailrc** to hold alias and feature settings set up by the user. Most single configuration files end in the letters **rc**. **ftp** uses a file called **.netrc**. Most newsreaders use a file called **.newsrc**. Entries in configuration files are usually set by the application, though you can usually make entries directly by editing the file. Applications have their own set of special variables to which you can define and assign values. Of particular interest is the **.wm_style** file that holds the name of the Window manager the user wants to use. You can edit and change the name there to that of another window manager to start up a new one. You can list the configuration files in your home directory with the **ls -a** command.

Chapter 14: Office Applications

Overview

A variety of office suites is now available for Linux including professional-level word processors, presentation managers, drawing tools, and spreadsheets. The freely available versions are described in this chapter. Currently, you can download personal (noncommercial) versions for both WordPerfect and StarOffice from the Internet for free. KOffice is an entirely free office suite for use with KDE. The Gnome Office is integrating Gnome applications into a productivity suite that will be freely available. Ximian is developing a professional-level office suite for Gnome. In addition, Sun has initiated the development of an open source Office suite using StarOffice code. The applications are known as OpenOffice and will provide Office applications integrated with Gnome. You can also purchase commercial office suites such as Applixware from Red Hat and Corel Office from Corel. *Applixware* includes a word processor, a spreadsheet, a presentation graphics tool, a drawing tool, an e-mail client, and an object-oriented application builder. Table 14-1 lists freely available Linux Office projects. Red Hat now includes AbiWord, a professional-level word processor.

Table 14-1: Linux Office Projects

Web Site	Description
koffice.kde.org	KOffice Suite, for KDE
www.gnome.org/gnome-office	Gnome Office, for Gnome
www.sun.com/staroffice	Star Office Suite
www.openoffice.org	OpenOffice open source office suite based on Star Office
www.ximian.com	Ximian Gnome desktop and office Applications
linux.corel.com	WordPerfect word processor

Accessibility to Microsoft Office

One of the primary concerns for new Linux users is what kind of access they would have to their Microsoft Office files, particularly Word files. The Linux operating system and many applications for it are designed to provide seamless access to MS Office files. The Intel version of Linux can directly mount and access any Windows partition and its files. The major Linux Office suites, including WordPerfect, KOffice, and particularly Star Office, all read and manage any Microsoft Office files. In addition, these office suites are fast approaching the same level of support for office tasks as found in Microsoft Office.

Note If you want to use any Windows application on Linux, one important alternative is the VMware virtual platform technology, a commercial package. With Vmware, you can run any Windows application directly on your Linux system. For more information, check the VMware Web site at www.vmware.com. The Wine project has also developed a Windows 3.1 and Win32 interface that works on X running on Unix and Linux systems. Check www.winehq.com for more details.

KOffice

KOffice is an integrated office suite for the KDE (K Desktop Environment) consisting of several office applications, including a word processor, a spreadsheet, and graphic applications. All applications are written for the KOM component model, which allows components from any one application to be used in another. This means you can embed a spreadsheet from KSpread or a drawing from Kontour in a KWord document. You can obtain more information about KOffice from the KOffice Web site at **koffice.kde.org**.

Currently, KOffice includes KSpread, KPresenter, Kontour, KFormula, KWord, Kugar, Kivio, and Krayon (see Table 14-2). KSpread is a spreadsheet, KPresenter is a presentation application, Kontour is a vector drawing program, KWord is a Publisher-like word processor, KCharts generate charts and diagrams, KFormula is a formula editor, and Krayon is a bitmap image editor. Kugar is a report generator, and Kivio creates flow charts.

Table 14-2: KOffice Applications

Application	Description
KSpread	Spreadsheet
KPresenter	Presentation program
Kontour	Vector drawing program

Table 14-2: KOffice Applications

Application	Description
KWord	Word processor (desktop publisher)
KFormula	Mathematical formula editor
KChart	Tool for drawing charts and diagrams
Kugar	Report generator
Krayon	An image manipulation program
Kivio	Flow chart generator and editor (similar to Vivio)

Note The publisher's edition of Red Hat does not contain KOffice. You can download a Red Hat RPM package version from the Red Hat FTP site at **ftp.redhat.com**, and install it on your system.

Embedded components support real-time updates. For example, if you use KChart to generate a chart in a KWord document using data in a KSpread spreadsheet and then change the selected data in the spreadsheet, the KChart automatically updates the chart in the KWord document. In effect, you are creating a compound document-one made up of several applications. This capability is implemented by the KDE component model known as KParts. KParts replaces K Object Model/OpenParts (KOM/OP) component model used in KDE version 1.0, which was based directly on CORBA. Like KOM/OP, KParts provides communication between distributed objects. In this respect, you can think of an application working also as server, providing to other applications the services it specializes in. A word processor, specializing in services like paragraph formatting or spell checking, could provide these services to all KOffice applications. In that way, other applications do not need to have their own text formatting functions written into it.

KParts is implemented with DCOP, the Desktop Communications Protocol. This is a very simple, small, and fast IPC/RPC mechanism for InterProcess Communication (IPC) and is based on the X Window System's ICE (Inter-client Exchange) protocol. KDE applications now use DCOP libraries to manage their communications with each other. DCOP makes development of KOffice applications much easier and more stable.

With KOffice, you create one kind of document rather than separate ones for different applications. The different applications become views of this document, adding their components to it. KWord sets up the publishing and word processing components, Kontour adds drawing components, while KSpread adds spreadsheet components. You use the appropriate application to view the different components in the single document. This means you can have separate windows open at the same time for different components of the document.

KSpread is the spreadsheet application, which incorporates the basic operations found in most spreadsheets, with formulas similar to those used in Excel (see Figure 14-1). You can extend KSpread capabilities with Python scripts. It supports features such as embedded buttons for customized functions, automatic completion for cell contents, and formatting options such as backgrounds, borders, and font styles. To generate a diagram using selected cells, select the Insert Diagram entry from the KSpread menu. This starts up KDiagram, which you then use to create the diagram-which is then embedded in the spreadsheet. You can also embed pictures or formulas using Krayon, Kontour, or KFormula.

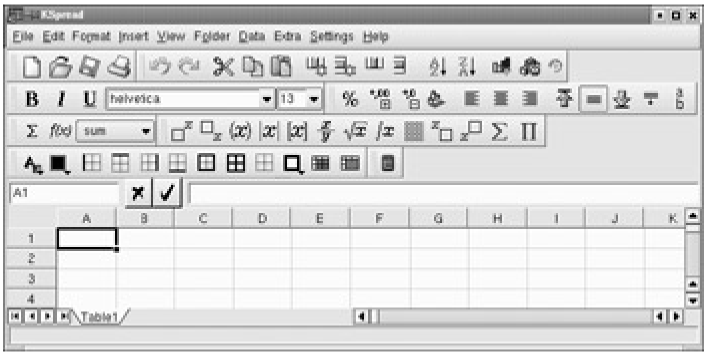

KSpread
File Edit Format Insert View Folder Data Extra Settings Help
helvetica
13
sum
A1
A B C D E F G H I J K
1
2
3
4
Table1

Figure 14-1: KSpread, KOffice spreadsheet

With KChart*,* you can create different kinds of charts, such as bar graphs, pie charts, and line graphs, as well as create diagrams (see Figure 14-2). To generate a chart, you can use data in KSpread to enter your data.

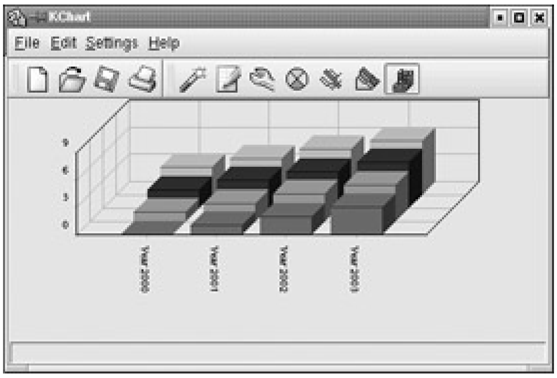


Figure 14-2: KChart, KOffice graphs

With KPresenter*,* you can create presentations consisting of text and graphics modeled using different fonts, orientations, and attributes such as colors (see Figure 14-3). You can add such elements as speech bubbles, arrows, and clip art, as well as embed any KOffice component. KPresenter supports standard editing operations such as shading, rotating, and coloring objects, as well as cut-and-paste and undo/redo capabilities. You can generate templates from a KPresenter document, in effect enabling you to use a document's configuration to create other documents. With KPresenter, you can also create special effects such as simple animation.

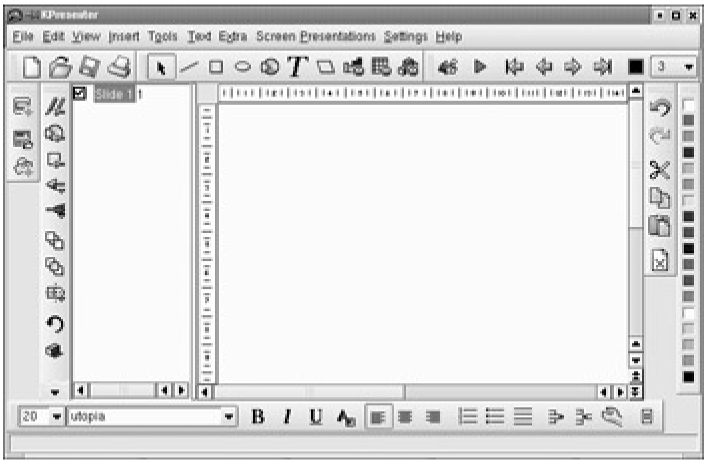

KPresenter
File Edit View Insert Tools Text Extra Screen Presentations Settings Help
3
Slide 1
20
utopia

Figure 14-3: KPresenter, KOffice presentation application

Kontour is a vector-based graphics program, much like Adobe Illustrator and Corel Draw (see Figure 14-4). It supports the standard graphic operations such as rotating, scaling, and aligning objects. Kontour also includes text formatting capabilities such as alignment to irregular boundaries. You can create complex illustrations using layers, using a layer manager to control the layers. Kontour also supports a number of import and export filters for image files of different types like **.jpeg**, **.gif**, and **.eps**.

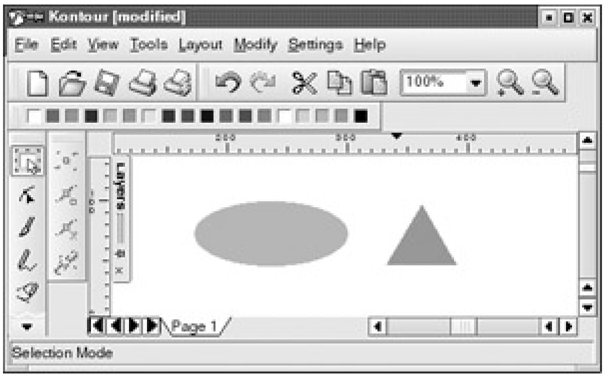
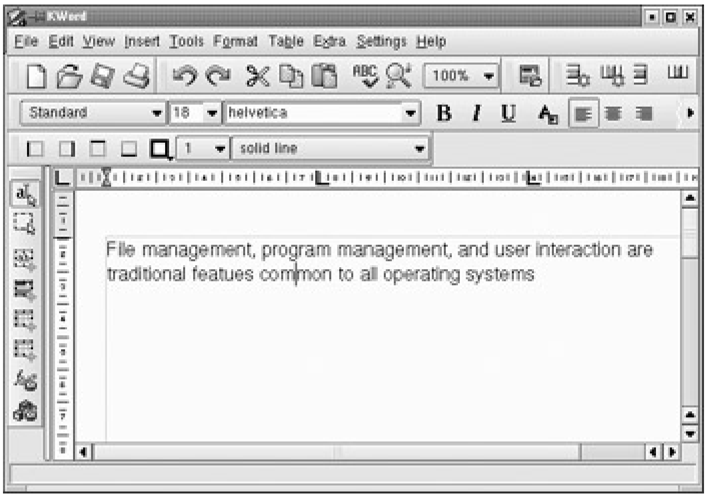


Figure 14-4: Kontour, KOffice illustration application

KWord can best be described as a desktop publisher, with many of the features found in publishing applications like Microsoft Publisher and FrameMaker (see Figure 14-5). Although it is also a fully functional word processor, KWord is not page-based like Word or WordPerfect. Instead, text is set up in frames that are placed on the page like objects. Frames, like objects in a drawing program, can be moved, resized, and even reoriented. You can organize frames into a frame set, having text flow from one to the other. Formatting can be applied to a frame set, changing features in all the frames belonging to it at once. The default frame set up for you when you first create a document is the same size as the page. This gives you the effect of a page-based word processor, enabling you to work as if you were using a standard word processor. You can, of course, change the size of your frame and add new ones, if you want.


KWord
File Edit View Insert Tools Format Table Extra Settings Help
100%
Standard
18
helvetica
B
I
U
1
solid line

File management, program management, and user interaction are traditional featues common to all operating systems

Figure 14-5: KWord, KOffice word processor and publisher

You can also insert images, illustrations, tables, and other KOffice components such as diagrams and spreadsheets. You can set up a frame to contain an image and place it on top of a text frame, configuring the text frame to flow its text around it.

KWord uses templates to set up a document. You have two different sets of templates from which to choose: one for desktop publishing (DTP) and the other for standard word processing (word processing). The desktop publishing templates enable you to move frames freely, whereas in the word processing templates, the frames are fixed to the size of the page.

KWord supports the standard word processing features for formatting paragraphs, text, and document elements, such as headers and footers, as well as lists and multiple columns. You can also define your own paragraph layouts, specifying features such as indentation, fonts, borders, and alignment. Layouts are the same as styles used in other word processors. Tables are implemented as frames, where each cell is its own frame.

KFormula is a formula editor used to generate mathematical formulas. Although KFormula does not have the power of TeX, you can use it to create fairly complex formulas. It supports standard components like roots, integral, and fractions, as well as fonts for Greek symbols.

Kivio is a flowchart application similar to Vivio. Kivio has the ability to generate flowcharts using scriptable objects. Given a network, Kivio can generate a flow chart for it. With Java header files it can generate a flow chart of Java objects. Krayon (formerly KImageShop) is an image editor, much like PhotoShop.

Kugar is a business-quality report generator based on XML that can embed reports in KOffice applications and even be displayed on Konqueror. Kugar supports page features like numbering, dates, headers, and footers; numeric operations like sums, averages, and currency formats; and word processing features like fonts, colors, and text formatting.

Gnome Office, OpenOffice, and Ximian

Office applications for Gnome have been developed independently, such as the Gnumeric spreadsheet. Currently, Gnome Office is an attempt to integrate the various office applications into a productivity suite. Although most are still under development, some have working stable versions you can download and install. You can find out more from the Gnome Office at **www.gnome.org/gnome-office**. Here, you can link to download pages for current versions and view screen shots. A current listing is shown in Table 14-3. All implement the CORBA model for embedding components, ensuring drag-and-drop capability throughout the Gnome interface. All are based on a set of Gnome technologies, including Bonobo, Gnome-Print, and XML. Bonobo is the Gnome architecture for supporting compound documents and reusable software components. Gnome-Print is the Gnome printing architecture designed to support graphics applications as well as work easily with any printing resource. XML is used as the native file format to support the easy exchange of data between Gnome applications.

Table 14-3: Gnome Office

Application	**Description**
Achtung	Presentation manager
AbiWord	Cross-platform word processor
Balsa	E-mail client
Gfax	Send and receive faxes
Galeon	Web browser

Table 14-3: Gnome Office

Application	**Description**
Gnumeric	Spreadsheet
GnuCash	Personal finance manager
Dia	Diagram and flow chart editor
Evolution	E-mail and calendar
Eye of Gnome	Image viewer
GIMP	GNU image manipulation program
Gnome-DB	Database connectivity
Gnumeric	Spreadsheet
Gnome Personal Information Manager	Calendar/organizer and an address book
Gnumeric	Spreadsheet
Guppi	Plotting and graphing program
Sketch	Vector drawing package
Toutdoux	Project manager

Currently, Gnome Office includes AbiWord, Gnumeric, GIMP, Dia, Eye of Gnome, Gnome-PIM, and Gnome-DB. AbiWord is a word processor, Gnumeric is a spreadsheet, the GIMP is the Gnome image editing application, Dia is a diagram composer, Eye of Gnome is an image viewer, the Gnome Personal Information Manager (Gnome-PIM) is a personal information manager, and Gnome-DB provides database connectivity (see Table 14-3).

The Gnumeric is the Gnome spreadsheet, a professional-level program meant to replace commercial spreadsheets (see Figure 14-6). Like Gnome, Gnumeric is freely available under the GNU Public License. Gnumeric is included with the Gnome release, and you will find it installed with Gnome on the Red Hat and SuSE distributions. Its development is currently managed by Ximian. You can download current versions from **www.ximian.com** or **www.gnome.org/projects/gnumeric**. Gnumeric supports standard GUI spreadsheet features, including autofilling and cell formatting, and it provides an extensive number of formats. It supports drag-and-drop operations, enabling you to select and then move or copy cells to another location. Gnumeric also supports plug-ins, making it possible to extend and customize its capabilities easily.

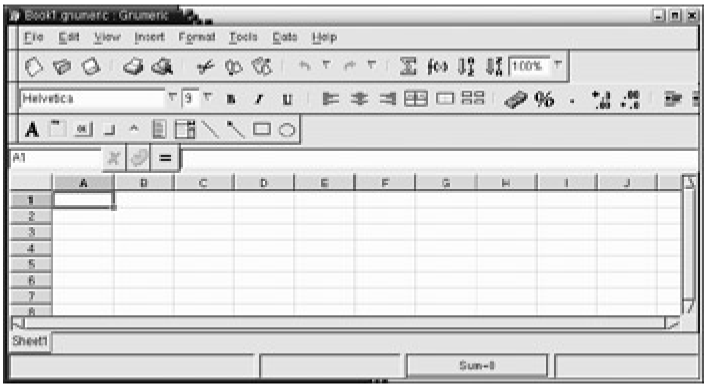
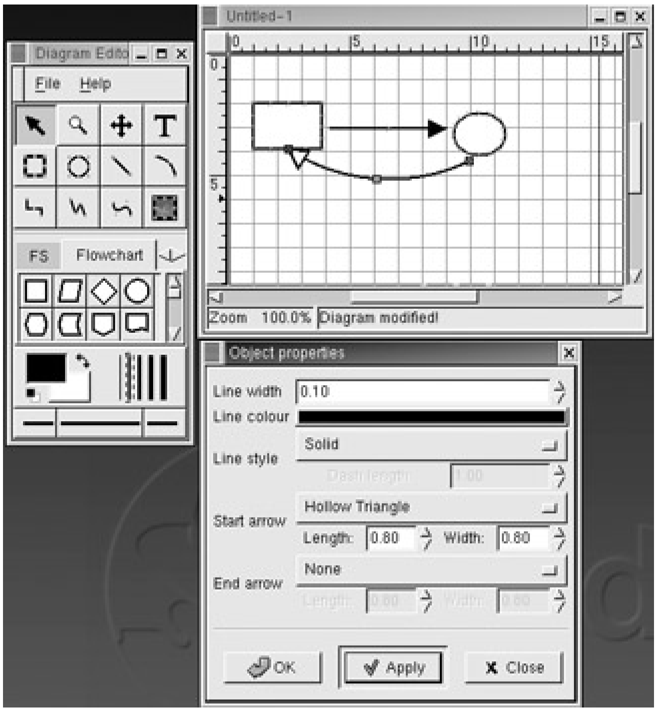

Book1.gnumeric : Gnumeric
File Edit View Insert Format Tools Data Help
100%
Helvetica
9
A1
A B C D E F G H I J
1 2 3 4 5 6 7 8
Sheet1
Sum=0

Figure 14-6: Gnumeric, Gnome spreadsheet

AbiWord is an open source word processor that supports such features as unlimited undo operations and Word97 imports. It is part of a set of desktop productivity applications being developed by the AbiSource project (**www.abisource.com**).

The GIMP is the GNU image manipulation program that supports operations such as photo retouching, image composition, and image authoring (see the section on graphics). You can obtain more information from **www.gimp.org**.

Dia is a drawing program designed to create diagrams (see Figure 14-7). You can select different kinds of diagrams to create, such as database, circuit object, flow chart, and network diagrams. You can easily create elements along with lines and arcs with different types of endpoints such as arrows or diamonds. Data can be saved in XML format, making it easily transportable to other applications.



Zoom 100.0% Diagram modified!

Line width 0.10

Line colour

Line style Solid

Dash length 1.00

Start arrow Hollow Triangle

Length: 0.80 Width: 0.80

End arrow None

Length: 0.80 Width: 0.80

Figure 14-7: Dia, Gnome diagrams

The Eye of Gnome (EOG) is an image viewer that can also function as an image cataloging program. It is designed to be both fast and provide high-quality image displays.

Gnome Personal Information Manager (Gnome-PIM) is a personal information manager that currently includes both a calendar (see Figure 14-8) and an address book. Gnome-PIM is included with the current distribution of Gnome. You can obtain updates from **www.gnome.org/gnome-office**.

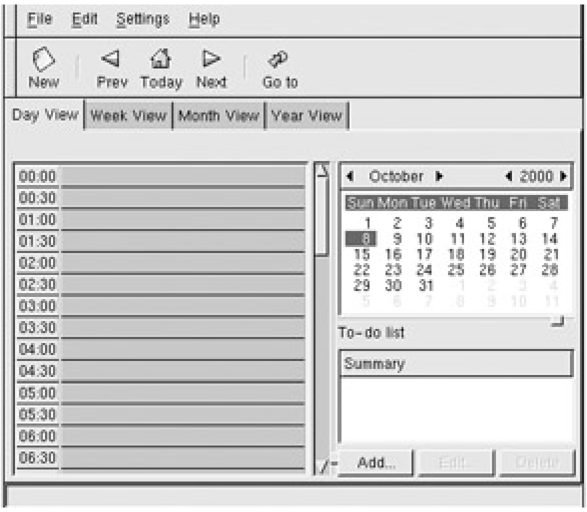


Figure 14-8: Gnome PIM calendar

Gnome-DB is a suite of libraries and applications that allow easy access to different database systems. It provides an API which databases can plug into. These backend connections are based on CORBA. Through this API, Gnome applications can then access a database. You can find out more about Gnome-DB at **www.gnome.org/gnome-db**.

Ximian is a Gnome project designed to provide a professional-level desktop for Gnome that will include a full range of office applications. You can find out more about Ximian at **www.ximian.com**. Ximian currently provides an enhanced version of the Gnome desktop called Ximian Gnome. It also supports development of the Gnumeric spreadsheet. Its first office application is a contact, calendar, and mail client application called Evolution (see Chapter 17). You can download Evolution from the Ximian Web site, but it currently requires the Ximian Gnome desktop to run. Their next project is a word processor.

The OpenOffice Suite of applications currently covers several core applications used in StarOffice. StarOffice code has been released as an open source project called OpenOffice.org, at **www.openoffice.org**. The open source versions of StarOffice applications, known as Open Office, are being integrated with Gnome. This, in effect, creates a Gnome OpenOffice. The applications currently under development are listed in Table 14-4. Figure 14-9 shows the four applications.

Table 14-4: Gnome Open Office

Application	Description
OpenCalc	OpenOffice spreadsheet
OpenDraw	OpenOffice drawing application
OpenWriter	OpenOffice word processor
Impress	OpenOffice presentation manager

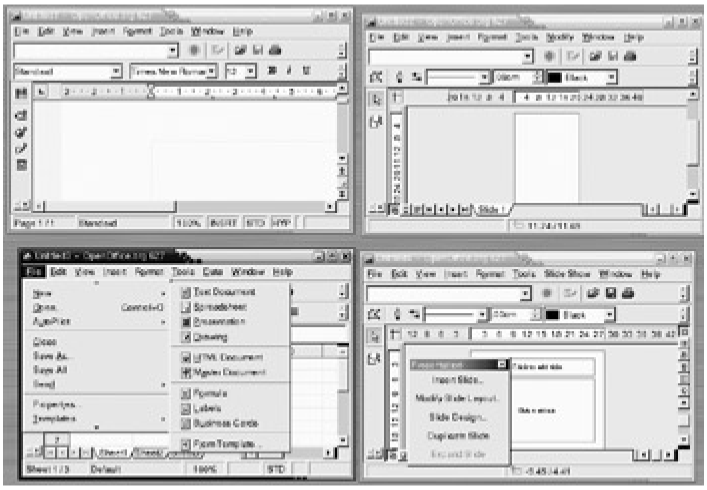


Figure 14-9: OpenOffice.org

Like StarOffice, OpenOffice is an integrated suite of applications. Initially the suite opens with the OpenWriter window. You can select New from the file menu and then select a different application (see Figure 14-9). The application will open in a separate window. OpenOffice is still very much in development, but holds the promise of providing professional-level office applications integrated into the Gnome desktop.

WordPerfect

The personal version of Corel's WordPerfect word processor is now available for Linux and is free (it is included with OpenLinux). The personal version is a fully functional word processor. However, it does not currently support TrueType fonts and it does not allow you to import other objects, such as images. You can download WordPerfect from the Corel Web site at **linux.corel.com**. WordPerfect is more than just a word processor. You can use it to create drawings, spreadsheets, and charts, as well as to edit and publish Web pages.

When you first start WordPerfect, a small window is displayed with the WordPerfect logo and four menus: Program, Preferences, Window, and Help. From the Program menu, you can select WordPerfect. In the Preferences menu, you can open a window with icons for configuring your printer, selecting fonts, choosing colors, and selecting conversion filters. The Window menu moves you to different open windows.

WordPerfect provides many of the standard word processing features, including cut-and-paste operations, font and paragraph styles, and document formatting. The extensive features for WordPerfect are indicated by its set of toolbars. WordPerfect includes such editing features as Grammar-As-You-Go, which checks and highlights suspicious phrases and offers suggestions. With Spell-As-You-Go, words are identified that might be misspelled as you type them. Corel Versions keeps track of document revisions for workgroup collaboration.

WordPerfect includes a chart and drawing tool for creating figures. You can perform drawing operations such as sizing and rotating images, as well as contouring text over image shapes. The drawing tool supports features such as gradients, patterns, and groupings. You can create a variety of different charts, including 3-D, area, and line charts. WordPerfect supports a number of spreadsheet functions with which you can create tables with spreadsheet cells. You can use such data to generate charts.

You can also use WordPerfect as a Web page editor and publisher, adding or changing HTML components. Use WordPerfect to create your HTML document with hyperlinks and bookmarks, and then place them on your Web site. Any text beginning with an Internet protocol, such as **www**, **ftp**, and **http**, is automatically set up as a hyperlink. You can also save Web pages as WordPerfect documents for easy editing.

Note WordPerfect supports an extensive number of file formats, including Microsoft Word files. With WordPerfect, you can effectively edit your Word files.

WordPerfect also has its own file manager. You can use the file manager to locate and open files, but it also performs other operations. You can create directories and modify file permissions, as well as move and copy files.

StarOffice

StarOffice is a fully integrated and Microsoft Office-compatible suite of office applications developed and supported by Sun Microsystems. It includes Web-enabled word processing, spreadsheet, presentation, e-mail, news, chart, and graphic applications (See Table 14-5). Versions of StarOffice exist for Linux, Windows, Mac, Solaris, and OS/2. With StarOffice, you can access Microsoft Office (including 2000) files and data to create spreadsheets, presentations, and word processing documents. You can save StarOffice documents in Microsoft formats or as HTML files that you can post on Web sites.

Table 14-5: StarOffice Applications

Application	**Description**
StarDesktop	Main desktop window for StarOffice applications
StarWriter	Word processor
StarImpress	Presentation manager
StarDraw	Drawing tool
StarChart	Chart and graph creator
StarMail	E-mail client
StarDiscussion	Newsgroup client
StarMath	Mathematical formulas
StarImage	Image editor
StarCalc	Spreadsheet
StarSchedule	Schedule manager
StarBase	Relational database

StarOffice is free for all noncommercial, private users, as well as students. You can download a free copy of StarOffice from the Sun Web site at **www.sun.com/products/staroffice**. The package is about 65 megabytes. The Web site also contains information such as online manuals and FAQs.

Note Development for StarOffice is being carried out as an open source project called openoffice.org. The core code for StarOffice is now open source, and anyone can download and use it. See the OpenOffice.org Web site at **www.openoffice.org** for more information. The code developed in the openoffice.org project will then be incorporated into future releases of StarOffice. Currently the word processor, presentation manager, drawing tool, chart tool, and spreadsheet are under open source development at **openoffice.org**.

StarOffice describes itself as implementing a task-oriented approach to office projects. You can complete an entire project using a variety of different tools in just one place, StarOffice. In addition, StarOffice applications are fully Internet-aware, enabling you to connect directly to Web sites and access information from your word processor, spreadsheet, or presenter.

When you start up StarOffice, you are presented with the StarOffice desktop window. From here, you can create documents and access other StarOffice applications. The left pane in this window is the Explorer. This is a tree menu that lists the different resources you can use, such as an address book, a gallery of clip art, and FTP server URLs. The main window of the StarOffice desktop shows icons for applications you can use, your StarOffice documents, and other tools.

You use StarOffice by creating projects. In a particular project, you can place resources such as images, Internet links, and e-mail messages, or office documents such as spreadsheets and word processing documents. To create a new document, click the New button in the bottom status bar and select the application you want to use from the pop-up menu. You can create spreadsheets, word processing documents, presentation files, mail messages, charts, images, mathematical formulas, and even Web pages (see Table 14-4). You can also create frame sets for use in Web page frames. Initially, windows are attached (docked) to the desktop window. You can unattach them to their own floating windows by double-clicking the gray area between them.

StarOffice has its own mail and news clients. With StarMail, you can define mail accounts and access your e-mail, as well as compose and send messages. With StarDiscussion, you can access newsgroups, saving articles and posting your own. You can set up icons and entries for mail accounts and newsgroups on your desktop and Explorer window.

The StarWriter word processor supports standard word processing features, such as cut and paste, spell checker, and text formatting, as well as paragraph styles. You can also insert objects in your text, such as images, diagrams, or text frames. Text can be configured to flow around them. A text frame can be further edited to create banner-like text, coloring, blending, and shaping text. The Navigator enables you to move through the document by page or by object, such as from one image to another.

You can open and save documents in the MS Office, WordPerfect, Lotus 1-2-3, and AmiPro formats. This means you can effectively edit MS Word documents with StarOffice. StarWriter also functions as a Web browser-capable of displaying HTML pages-and supports Java, JavaScript, Navigator, and Explorer plug-ins. At the same time, you can edit a Web page, turning StarWriter into a Web page editor.

You can embed objects within documents, such as using StarChart to create a bar chart using data in the spreadsheet. With StarMath, you can create formulas that you can then embed in a text document.

With the presentation manager (StarImpress), you can create images for presentations, such as circles, rectangles, and connecting elements like arrows, as well as vector-based illustrations. StarImpress supports advanced features like morphing objects, grouping objects, and defining gradients. You can also create animation effects and use layers to generate complex images. Any components from other StarOffice applications can be embedded in a presentation document. You can also import Microsoft PowerPoint files and save presentation files as HTML files. An AutoPilot Wizard for StarImpress walks you through the steps for creating a presentation.

StarDraw is a sophisticated drawing tool that includes 3-D modeling tools. You can create simple or complex images, including animation text aligned on curves. You can use it to create buttons and icons for your Web page.

StarSchedule provides scheduling and task management that can be used to coordinate efforts by a group of users. You can use it to track events and to-do lists, connecting automatically to the address book. StarSchedule includes a reminder system to display pop-up alerts and to send e-mail reminders. The StarSchedule server operates independently from StarOffice to provide scheduling services to any client on your network. StarBase is a relational database, somewhat like MS Access, which supports drag-and-drop operations for importing data to your StarOffice applications.

Chapter 15: Database Management Systems, Graphics Tools, and Multimedia

Overview

A variety of database management systems is now available for Linux including high-powered, commercial-level database management systems, such as Oracle, IBM, and Sybase. Most of the database management systems available for Linux are designed to support large relational databases. For small personal databases, you can use the desktop database management systems being developed for KDE and Gnome such as Gaby (see Chapter 20). In addition, some software is available for databases accessed with the Xbase database programming language. These are smaller databases using formats originally developed for dBase on the PC. Various database management systems available to run under Linux are listed in Table 15-1.

Table 15-1: Database Management Systems for Linux

System	**Site**
Oracle	Oracle database **www.oracle.com**
Sybase	Sybase database **www.sybase.com**
DB2	IBM database **www.software.ibm.com/data/db2/linux**
Informix	Informix database **www.informix.com/linux**
Adabas D	Adabas D database **www.adabas.com**
MySQL	MySQL database **www.mysql.com**
GNU SQL	The GNU SQL database **www.ispras.ru/~kml/gss**

Table 15-1: Database Management Systems for Linux

System	Site
PostgreSQL	The PostgreSQL database **www.postgresql.org**
Flagship	Interface for Xbase database files **www.fship.com/free.html**
Gaby	Gnome desktop personal database **gaby.sourceforge.net**.

You can also use a wide range of graphics tools, ranging from simple image viewers like KImage to sophisticated image manipulation programs like the GIMP. You also have newer graphics tools for Gnome and KDE desktops, as well as the older X Window System, from which to choose. Graphics tools available for use under Linux are listed in Table 15-2.

Table 15-2: Graphics Tools for Linux

KDE	Description
Kview	Simple image viewer for GIF and JPEG image files
Ksnapshot	Screen grabber
Kfourier	Image processing tool that uses the Fourier transform
KuickShow	Image browser and viewer
Kshow	Simple image viewer
Kpaint	Paint program
Krayon	Image editor
Gnome	
Gqview	Image viewer
ImageShaker	Digital image processing
GIMP	GNU Image Manipulation Program
Electric Eyes	Image viewer
GPhoto	Digital Camera application
X Window System	
xv	Screen grabber and image conversion
Xpaint	Paint program
Xfig	Drawing program
Xmorph	Morphs images
Xfractals	Generates fractal images

There is also strong support for multimedia tasks from video and DVD to sound and music editing (see Table 15-3).

Table 15-3: Multimedia Applications

Application	Description
GTV	Gnome MPEG video player
Xine	Multimedia player for video, DVD, and audio

Table 15-3: Multimedia Applications

Application	Description
Gnome-tv	TV tuner
aKtion	KDE video player
kscd	Music CD player
Knapster2	Napster-like MP3 download utility
XMMS	MP3 and CD Player
Xplaycd	Music CD player
Xanim	Animation and video player
mpg123	Command line MP3 player
RealPlayer	RealMedia and RealAudio streaming media (**www.real.com**)
KreateCD	KDE CD writing interface for cdrecord, mkisofs, and cdda2wav
CD-Rchive	KDE CD burner and ripper
Noatun	KDE multimedia player

Database Management Systems

Database software can be generally organized into three categories: SQL, Xbase, and desktop databases. *SQL-based databases* are professional-level relational databases whose files are managed by a central database server program. Applications that use the database do not access the files directly. Instead, they send requests to the database server, which then performs the actual access. *SQL* is the query language used on these industrial-strength databases.

The *Xbase language* is an enhanced version of the dBase programming language used to access database files whose formats were originally developed for dBase on the PC. With Xbase, database management systems can directly access the database files. Xbase is used mainly for smaller personal databases, with database files often located on a user's own system.

Desktop databases are being developed for both Gnome and KDE. Currently, these are personal databases designed for individual users. Gaby is meant to be used for a user's personal records and is designed with a plug-in structure that can easily extend its capabilities.

SQL Databases (RDMS)

SQL databases are relational database management systems (RDMSs) designed for extensive database management tasks. Many of the major SQL databases now have Linux versions, including Oracle, Informix, Sybase, and IBM (but not, of course, Microsoft). These are commercial and professional database management systems of the highest order. Linux has proved itself capable of supporting complex and demanding database management tasks. In addition, many free SQL databases are available for Linux that offer much the same functionality. Most commercial databases also provide free personal versions, as do Oracle, Adabas D, and MySQL.

PostgreSQL

PostgreSQL is based on the POSTGRES database management system, though it uses SQL as its query language. POSTGRES is a next-generation research prototype developed at the University of California, Berkeley. Linux versions of PostgreSQL are included in Red Hat, Debian, and Slackware distributions. Your Red Hat Linux CD-ROM includes PostgreSQL. Updates can be obtained from the Red Hat update site at **ftp.redhat.com** or by using the Red Hat Network to update your system. You can also download current versions from the PostgreSQL Web site at **www.postgresql.org**. Development is being managed by a team of developers over the Internet. Red Hat now offers a version of Linux with PostgresSQL, called the Red Hat Database, that is optimized for database operations.

PostgresSQL is often used to provide database support for Internet servers with heavy demand such as Web servers. With a few simple commands you can create relational database tables. Use the **createuser** command to create a PostgresSQL user that you can then log into the server with. You can then create a database with the **createdb** command and construct relational tables using the create table directive. With an **insert** command you can add records and then view them with the **select** command. Access to the server by remote users is controlled by entries in the **pg_hba.conf** file located in PostgresSQL directory, usually **/var/lib/pgsql**.

Oracle

Oracle offers a fully functional version of its Oracle8i (soon to be superseded by Oracle9i) database management system for Linux, as well as the Oracle Application Server. You can download trial versions from the Oracle Web site at **www.oracle.com**. Oracle8i is a professional database for large databases specifically designed for Internet e-business tasks. Expect to use a gigabyte of memory just to install it. The Oracle Application Server provides support for real-time and commerce applications on the Web. As Linux is a fully functional version of Unix, Oracle is particularly effective on it. Oracle was originally designed to operate on Unix, and Linux is a far better platform for it than other PC operating systems.

Oracle offers extensive documentation for its Linux version that you can download from its Documentation page, to which you can link from the Support pages on its Web site. The documentation available includes an installation guide, an administrator's reference, and release notes, as well as the generic documentation. You can find specific information on installing and configuring Oracle for Linux in the Oracle Database HOW-TO.

Informix

Informix offers an integrated platform of Internet-based applications called Informix Internet Foundation.2000 on Linux. These include the Informix Dynamic Server, their database server. Informix Dynamic Server features Dynamic Scalable Architecture, making it capable of effectively using any hardware setup. Informix only provides commercial products. No free versions exist, though the company currently provides special promotions for Linux products. You can find out more about Informix at **www.informix.com/linux**.

Informix strongly supports Linux development of its Informix line. It provides developer support through its Informix Developer Network (**www.informix.com/idn**). Informix has a close working relationship with Red Hat and plans to provide joint technical support for their products.

Sybase

For Linux, Sybase offers the Sybase Adaptive Server Enterprise server (see **www.sybase.com**). You can currently download the Adaptive Server Enterprise server from the Web page **www.sybase.com/products/linux/**. The Sybase Enterprise database features data integration that coordinates all information resources on a network. SQL Anywhere is a database system designed for smaller databases, though with the same level of complexity found in larger databases.

DB2

IBM provides a Linux version of its DB2 Universal Database software. You can download it free from the IBM DB2 Web page for Linux, **www.software.ibm.com/ data/db2/linux/**. DB2 Universal Database for Linux includes Internet functionality along with support for Java and Perl. With the Web Control Center, administrators can maintain databases from a Web browser. DB2 features scalability to expand the database easily, support for Binary Large Objects, and cost-based optimization for fast access. DB2 is still very much a mainframe database, though IBM is currently working on refining its Unix/Linux version.

Adabas D

Adabas D is an intermediate relational database, not quite as powerful or as large as Oracle, which is meant for use on smaller networks of personal databases. It still provides the flexibility and power found in all relational databases. Adabas D provides a personal version for Linux free. For commercial uses, you have to purchase a copy. You can also check the Adabas D Web site at **www.adabas.com**.

MySQL

MySQL is a true multiuser, multithreaded SQL database server, supported by MySQL AB. MySQL is an open source product available free under the GPL license. You can download a copy from its Web site at **www.mysql.com**. The site also includes detailed documentation, including manuals and FAQs. RPM packages for Red Hat are included on Red Hat releases.

MySQL is structured on a client/server model with a server daemon (**mysqld**) filling requests from client programs. MySQL is designed for speed, reliability, and ease of use. It is meant to be a fast database management system for large databases and, at the same time, reliable with intensive use.

GNU SQL

GNU SQL is the GNU relational database developed by a group at the Institute for System Programming of the Russian Academy of Sciences and supported by the GNU organization. It is a portable multiuser database management system with a client/server structure that supports SQL. The server processes requests and performs basic administrative operations, such as unloading parts of the database used infrequently. The clients can reside on any computer of a local network. GNU SQL uses a dialect of SQL based on the SQL-89 standard and is designed for use on a Unix-like environment. You can download the database software from the GNU FTP site at **ftp.gnu.org**. For more information, contact the GNU SQL Web site at **www.ispras.ru/~kml/gss**.

Xbase Databases

Databases accessed with Xbase are smaller in scale, designed for small networks or for personal use. Many are originally PC database programs, such as dBaseIII, Clipper, FoxPro, and Quicksilver. Currently, only Flagship provides an interface for accessing Xbase database files, although the Harmony project is currently developing a Clipper clone that may run on Linux systems.

Flagship is a compiler with which you can create interfaces for querying Xbase database files. The interfaces support menus and dialog boxes, and they have function calls that execute certain database queries. Flagship can compile dBaseIII+ code and up. It is compatible with dBase and Clipper and can access most Xbase file formats, such as **.dbf**, **.dbt**, **.fmt**, and **.frm**. One of Flagship's key features is that its interfaces can be attached to a Web page, enabling users to update databases. Flagship is commercial software, though you can download a free personal version from its Web site at **www.fship.com/free.html**.

Desktop Database

Both Gnome and KDE also have database management applications that take advantage of their respective desktops. KDE has several front-end interfaces for standard databases. Gnome's Gaby takes advantage of the interface to display and search user databases.

KDE: Rekall

KDE provides several front-end interface applications for many of the popular databases used on Linux. MySQL Navigator, VMySQL, and SqlGui are front ends for the MySQL database. KSql, KPGsql, and KPSql are front ends for Postgresql; KOra is a front end for Oracle.

Rekall is a database interface designed by TheKompany.com for KDE that uses KDE-DB to enable you to create simple databases that can be implemented on high-powered systems like MySQL, Postgresql, and Oracle. With Rekall, you can design a personal database much like you would with db2 or MS Access, and then use the KDE-DB layer to implement this database on a complex system. This gives you the simplicity of a personal database combined with features of a fully functional RDMS database, providing capabilities like multiuser access. You can find out more about Rekall at **www.theKompany.com**.

In addition, several specialized database management systems are available. Kaspaliste and KBiblio are powerful bibliographic database systems, and Kmp3DB and MP3Organizer let you index and search MP3 files.

Gaby

Gaby is a small personal database manager using GTK+ and Gnome. It provides access to user databases, such as those for addresses, books, and even photos. Gaby's plug-in design makes it easily extensible. You can find out more about Gaby at **gaby.sourceforge.net**. Its interface has the same browser-like tools found in most Gnome applications. You can download the current version of Gaby from the Gnome software map at **www.gnome.org**.

Graphics Tools

Gnome, KDE, and the X Window System support an impressive number of graphics tools, including image viewers, window grabbers, image editors, and paint tools. On the KDE and Gnome desktops, these tools can be found under either a Graphics submenu or the Utilities menu.

KDE Graphics Tools

The kview program is a simple image viewer for GIF and JPEG image files. The ksnapshot program is a simple screen grabber for KDE, which currently supports only a few image formats. The kfourier program is an image-processing tool that uses the Fourier transform to apply several filters to an image at once. The kshow program is a simple image viewer. The kuickShow program is an easy-to-use, comfortable image browser and viewer, based on imlib. The kpaint program is a simple paint program with brushes, shapes, and color effects. Krayon is a professional image paint and editing application that is part of KOffice 2.1 (formerly KImageShop).

Gnome Graphics Tools

ImageShaker is a digital image-processing tool that includes an extensive set of graphics filters, such as alpha blending and median. It uses a stream-like approach that allows batch processing.

Electric Eyes is a simple image viewer. Right-click its window to display a pop-up menu with options. You can load images and also move back and forth through previously viewed ones. You can even make an image your background.

GQview is a simple image viewer supporting features like click file viewing, thumbnail preview, zoom, drag and drop, and external editor support, as well as slideshow and full-screen options. Multiple files can selected for moving, copying, deleting, renaming, or dragging. See **gqview.sourceforge.net** for more information.

The GIMP is the GNU Image Manipulation Program*,* a sophisticated image application much like Adobe Photoshop. You can use GIMP for such tasks as photo retouching, image composition, and image authoring. It supports features like layers, channels, blends, and gradients. GIMP makes particular use of the GTK+ widget set. You can find out more about the GIMP from its Web site at **www.gimp.org**. You can also download the newest versions from here. GIMP is freely distributed under the GNU Public License.

The gPhoto program is a digital camera tool that can load, select, and edit photos from a digital camera connected to your system. With gPhoto, you can generate thumbnail images to let you select and organize your photos. You can also process photos, changing their orientation.

X Graphic

The xv program is a screen capture and image-editing program. After displaying the main screen, right-click it to display its control screen. Use the Grab button to scan a window or a section you select with a drag operation using your middle mouse button. Once you have scanned a window or screen section, you can crop it with a drag operation with your left mouse key and then use crop to reduce it to that selected section. With xv, you can also convert an image file from one format to another. Just load the image and then save it as another file using a different image format.

The xpaint program is a painting program, much like MacPaint. You can load paint pictures or photographs, and then create shapes, add text, and add colors. You can use brush tools with various sizes and colors. The xfig program is a drawing program, and xmorph enables you to morph images, changing their shapes.

Multimedia

Many applications are available for both video and sound, including sound editors, MP3 players, and video players (see Table 15-3). Linux sound applications include mixers, digital audio tools, CD audio writers, MP3 players, and network audio support. The Linux Midi and Sound pages currently at **www.xdt.com/ar/linux-snd** hold links to Web and FTP sites for many of these applications. Many sound applications are currently under development for Gnome, including sound editors, MP3 players, and audio players. Check the software map at **www.gnome.org** for current releases. A variety of applications is available for KDE, including a media player (kaimain), a mixer (kmix), an MP3 player (KJukeBoxMgr), a CD player (kscd), and even a Napster-like MP3 download utility (KNapster2). Check **apps.kde.com** for recent additions. Several X Window System-based mulitimedia applications are installed with most distributions such as Red Hat. These include XMMS, an MP3 and CD player, Xplaycd, a CD music player, and Xanim, an animation and video player. Several are included in standard Red Hat installations, and RPM packages for most can be obtained from Red Hat (**ftp.redhat.com**). The Open Sound System (OSS) site provides an extensive listing of available multimedia software at **www.opensound.com/ossapps.html.** Here you can find digital audio players, mixers, MP3 and MPEG players, and even speech tools. You can also download a copy of RealPlayer, the Internet streaming media player, from **www.real.com**. Be sure to choose RealPlayer for Unix, and select as your OS, Linux 2.x (libc i386) RPM.

Note Several CD Write programs that can be used for CD Music and MP3 writing (burners and rippers) are available from **apps.kde.com**. These include KreateCD, CD-Rchive, and KOnCD (Gnome software is under development). All use mkisofs, cdrecord, and cdda2wav CD writing programs, which are installed by Red Hat. You can download, compile, and install them on Red Hat. Make sure that any CD-R, CD-RW, and CD-ROM drives that are IDE drives are installed as SCSI drives (see Chapter 4).

Several projects are under way to provide TV, video, and DVD support for Linux (see Table 15-4). The site **linuxtv.org** provides detailed links to DVD, digital video broadcasting (DVB), and multicasting. The site also provides downloads of many Linux video applications. For DVD, the Linux Video and DVD Project (LiViD) at **www.linuxvideo.org** supports the development of MPEG 2 (DVD) software. LiViD DVD and multimedia players are currently under development. Information about recent efforts to develop Linux DVD can be had at www.opendvd.org. Xine is a multipurpose video player for Linux/Unix systems that can play video, DVD, and audio disks. See xine.sourgeforge.net for more information.

Table 15-4: Linux Multimedia Projects

Projects	Description
Linux MIDI and Sound Pages	Information and links to Linux Sound projects and site: www.xdt.com/ar/linux-snd
Advanced Linux Sound Architecture (ALSA)	The Advanced Linux Sound Architecture project (ALSA) is developed on Linux under the GPL: www.alsa-project.org
Open Sound System	Open Sound System. Extensive software links www.opensound.com
linuxtv.org	Links to Video, TV, and DVD sites
LiViD	The Linux Video and DVD Project www.linuxvideo.org
LinuxDVD	The LinuxDVD Project linuxdvd.corepower.com
GATOS	The Gneral ATI TV and Overlay Software www.linuxvideo.org/gatos
Xine	Xine Video player xine.sourceforge.net

For KDE, several video applications are available or currently under development, including video players (aKtion and Noatun). Check apps.kde.com for downloads. Currently available or under development for Gnome are TV tuners (Gnomevision and Gnome-tv), a video player (Gnome-Video), and a video editor (trinity). Red Hat currently installs GTV, an MPEG viewer. Check www.gnome.org.

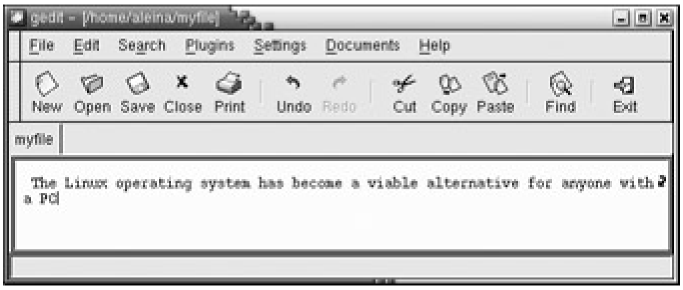
Chapter 16: Editors

Overview

Red Hat Linux includes several text editors. These range from simple text editors for simple notes to editors with more complex features such as spell-checkers, buffers, or pattern matching. All generate character text files and can be used to edit any Linux text files. Text editors are often used in system administration tasks to change or add entries in Linux configuration files found in the **/etc** directory or a user's initialization or application dot files located in a user's home directory. You can use any text editor to work on source code files for any of the programming languages or shell program scripts.

Note Red Hat now includes a very easy-to-use GUI-based text editor called Nedit. You can access it on both Gnome and KDE desktops.

Traditionally, most Linux distributions, including Red Hat, install the cursor-based editors Vim and Emacs. *Vim* is an enhanced version of the Vi text editor used on the Unix system. These editors use simple, cursor-based operations to give you a full-screen format. You can start these editors from the shell command line without any kind of X Window System support. In this mode, their cursor-based operations do not have the ease of use normally found in window-based editors. There are no menus, scroll bars, or mouse-click features. However, the K Desktop and Gnome do support powerful GUI text editors with all these features. These editors operate much more like those found on Mac and Windows systems. They have full mouse support, scroll bars, and menus. You may find them much easier to use than the Vi and Emacs editors. These editors operate from their respective desktops, requiring you first have either KDE or Gnome installed, though the editors can run on either desktop. Vi and Emacs, on the other hand, have powerful editing features that have been refined over the years. Emacs, in particular, is extensible to a full-development environment for programming new applications. Newer versions of Emacs, such as GNU Emacs and XEmacs, provide X Window System support with mouse, menu, and window operations. They can run on any window manager or desktop. In addition, the gvim version of the Vim editor also provides basic window operations. Table 16-1 lists several GUI based editors for Linux.



The Linux operating system has become a viable alternative for anyone with a PC

Figure 16-1: The gedit Gnome editor

Note Red Hat Linux includes in its distribution two fully functional word processors, KWord and Abiword. You can find out more on Abiword at www.abiword.com.

Gnome Editor: gedit

gedit is a basic text editor for the Gnome desktop (see Figure 16-1). It provides full mouse support, implementing standard GUI operations, such as cut-and-paste to move text, and click-and-drag to select text. It supports standard text editing operations such as Find and Replace. You can use gedit to create and modify your text files, including as configuration files. gedit also provides more advanced features such as print preview and configurable levels of undo/redo operations, and can read data from pipes. It features a plug-in menu that provides added functionality, and it includes plug-ins for spell-checking, encryption, e-mail, and text-based Web page display.

Table 16-1: Desktop Editors

The K Desktop	**Description**
KEdit	Text editor
Kate	Text and program editor, default for desktop
KJots	Notebook editor
KWord	Desktop publisher, part of KOffice
Gnome	
gedit	Text editor
Abiword	Word processor

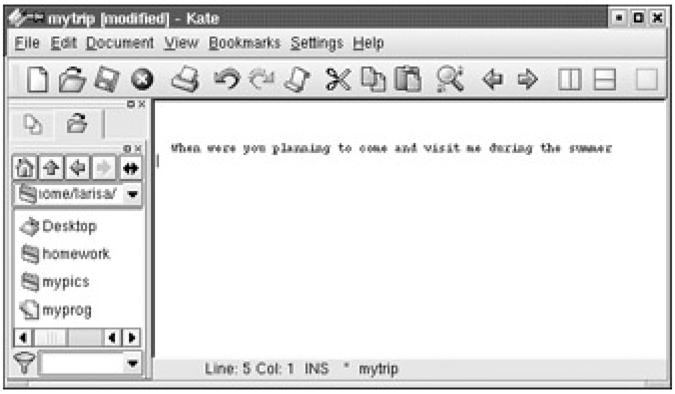
Table 16-1: Desktop Editors

The K Desktop	Description
X Windows	
GNU Emacs	Emacs editor with X Window System support
XEmacs	X Window System version of Emacs editor
gvim	Vim version with X Window System support
WordPerfect	Word processor that can edit text files
Abiword	Word processor
Nedit	GUI-based text editor

K Desktop Editors: KEdit, Kate, Kjots, and KWord

All the K Desktop editors provide full mouse support, implementing standard GUI operations, such as cut-and-paste to move text, and click-and-drag to select text. KEdit is a simple text editor meant for editing simple text files such as configuration files. A toolbar of buttons at the top of the KEdit window enables you to execute common editing commands easily using just a mouse click. With KEdit, you can also mail files you are editing over a network. The entry for KEdit in the K menu is listed simply as Text Editor. You can start up KEdit by entering the **kedit** command in a terminal window. The KOffice Office Suite also includes a word processor called KWord, which is a high-powered word processor you can also use as a simple editor.

A more advanced editor is Kate, with such features as spell checking, font selection, and highlighting (see Figure 16-2). Most commands can be selected using menus. A toolbar of icons for common operations is displayed across the top of the Kate window. A sidebar displays panels for a file selector and file list. With the file selector you can navigate through the file system, selecting files to work on. Kate also supports multiple views of a document, letting you display segments in their own windows, vertically or horizontally. You can also open several documents at the same time, moving between them with a file list.



When were you planning to come and visit me during the summer

Line: 5 Col: 1 INS * mytrip

Figure 16-2: Kate KDE editor

Kate is designed to be a program editor for editing software programming/ development-related source code files. Although Kate does not have all the features of Emacs or Vi, it can handle most major tasks. Kate can format the syntax for different programming languages, such as C, Perl, Java, and XML. In addition, Kate also has the capability to access and edit files on an FTP or Web site.

The editor KJots is designed to enable you to jot down notes in a notebook. It organizes notes you write into notebooks, called simply *books*. You can select the one you want to view or add to from the Books menu. To start KJots, select its entry in the Utilities menu or enter the **kjots** command in a terminal window.

Part IV: Red Hat Network Workstation

Chapter List

Chapter 17: Mail Clients

Overview

Your Linux system has electronic mail clients that enable you to send messages to other users on your system or other systems, such as those on the Internet. You can send and receive messages in a variety of ways, depending on the type of mail client you use. Although all electronic mail utilities perform the same basic tasks of receiving and sending messages, they tend to have different interfaces. Some mail clients operate on a desktop, such as KDE or Gnome. Others run on any X Windows window managers. Several popular mail clients were designed to use a screen-based interface and can run from only the command line. Other traditional mail clients were developed for just the command line interface, which requires you to type your commands. Most mail clients described here are included in standard Linux distributions and come in a standard RPM package for easy installation. For Web-based Internet mail services, such as Hotmail, Lycos, and Yahoo, you use a Web browser instead of a mail client to access mail accounts provided by those services. Table 17-1 lists several popular Linux mail clients.

Table 17-1: Linux Mail Clients

Mail Client	**Description**
KMail	The K Desktop mail client
Evolution	Ximian Gnome mail client
Balsa, Gmail, GNOMail, etc.	Gnome mail clients (see Table 17-2)
Mozilla Mail	Web browser-based mail client
Netscape	Web browser-based mail client
exmh	X Windows Mail Hander mail client
GNUEmacs and XEmacs	Emacs mail clients
Pine	Mail client and newsreader
Mutt	Screen-based mail client
Elm	Screen-based mail client

Table 17-1: Linux Mail Clients

Mail Client	Description
Mail	Original Unix-based command line mail client
nmh	New Mail Handler command line mail client

Table 17-2: Gnome Mail Clients

Application	Description
Balsa	Balsa is an e-mail client for GNOME that supports POP3, IMAP, local folders, and multithreading.
Evolution	Evolution is a Ximian integrated mail client, calendar, and contact manager.
Eucalyptus	Eucalyptus is an advanced MIME-compliant e-mail application.
Glacier	Glacier is an e-mail client for Gnome that supports mime parsing.
Gmail	Gmail is an experimental folder-based e-mail system (using mysql).
GnoMail	GnoMail is a mail user agent that uses the gnome-mailer interface.
LinPopUp	LinPopUp is an X Window graphical port of WinPopUp, running over Samba.
Mahogany	Mahogany is a cross-platform e-mail application.
N-tool	N-tool is a GUI mail tool that supports the Japanese language (ISO-2022-JP).
Pygmy	Pygmy is a Gnome mail client written in the Python programming language.
Spruce	Spruce is an e-mail client with support for multiple accounts.
Grin	Grin is a mail client with news reading features.
MMC	MMC is a Gnome mail client.
CSC Mail	CSC Mail is a mail client with support for multiple POP3 accounts and messaging.
Sonicmail	Sonicmail is a Gnome POP3 mail notifier applet.

Mail is transported to and from destinations using mail transport agents. Sendmail and Smail send and receive mail from destinations on the Internet or at other sites on a network (see Chapter 27). To send mail over the Internet, they use the Simple Mail Transport Protocol (SMTP). Most Linux distributions, including Red Hat, automatically install and configure Sendmail for you. On starting up your system, you can send and receive messages over the Internet.

Local and Internet Addresses

Each user on a Linux system has a mail address, and whenever you send mail, you are required to provide the address of the user to whom you are sending the message. For users on your local Linux system, addresses can consist of only the user's login name. When sending messages to users on other systems, however, you need to know not only the login name, but also the address of the system they are on. Internet addresses require the system address to be uniquely identified.

Most systems have Internet addresses you can use to send mail. Internet addresses use a form of addressing called *domain addressing.* A system is assigned a domain name, which when combined with the system name, gives the system a unique address. This domain name is separated from the system name by a period and may be further qualified by additional domain names. Here is the syntax for domain addresses:

```
login-name@system-name.domain-name
```

Systems that are part of a local network are often given the same domain name. The domain name for both the **garnet** and **violet** systems at UC Berkeley is **berkeley.edu**. To send a message to **chris** on the **garnet** system, you simply include the domain name:

```
chris@garnet.berkeley.edu.
```

In the next example, a message is sent to **chris,** located on the garnet system, using domain addressing:

```
$ mail chris@garnet.berkeley.edu < mydata
```

Early domain names reflect that the Internet was first developed in the United States. They qualify Internet addresses by category, such as commercial, military, or educational systems. The domain name **.com** indicates a commercial organization, whereas **.edu** is used for educational institutions. As the Internet developed into a global network, a set of international domain names was established. These domain names indicate the country in which a system is located-for example, **.fr** represents France, **.jp** represents Japan, and **.us** represents the United States.

Signature Files: .signature

You can end your e-mail message with the same standard signature information, such as your name, Internet address or addresses, or farewell phrase. Having your signature information automatically added to your messages is helpful. To do so, you need to create a signature file in your home directory and enter your signature information in it. A *signature file* is a standard text file you can edit using any text editor. Mail clients such as KMail enable you to specify a file to function as your signature file. Others, such as Mail, expect the signature file to be named **.signature**.

MIME

MIME (the term stands for Multipurpose Internet Mail Extensions) is used to enable mail clients to send and receive multimedia files and files using different character sets such as those for different languages. Multimedia files can be images, sound, or even video. Mail clients that support MIME can send binary files automatically as attachments to messages. MIME-capable mail clients maintain a file called **mailcap** that maps different types of MIME messages to applications on your system that can view or display them. For example, an image file will be mapped to an application that can display images. Your mail clients can then run that program to display the image message. A sound file will be mapped to an application that can play sound files on your speakers. Most mail clients have MIME capabilities built in and use their own version of the **mailcap** file. Others, like Elm, use a program called metamail that adds Pine support. MIME is not only used in mail clients. As noted in Chapters 8 and 9, both the KDE and Gnome file managers use MIME to map a file to a particular application so that you can launch the application directly from the file.

Applications are associated with binary files by means of the **mailcap** and **mime.types** files. The **mime.types** file defines different MIME types, associating a MIME type with a certain application. The **mailcap** file then associates each MIME type with a specified application. Your system maintains its own MIME types file, usually **/etc/mime.types**.

Entries in the MIME types file associate a MIME type and possible subtype of an application with a set of possible file extensions used for files that run on a given kind of application. The MIME type is usually further qualified by a subtype, separated from the major type by a slash. For example, a MIME type image can have several subtypes such as jpeg, gif, or tiff. A sample MIME type entry defining a MIME type for JPEG files is shown here. The MIME type is image/jpeg, and the list of possible file extensions is "jpeg jpg jpe."

```
image/jpeg jpep jpg jpe
```

The applications specified will depend on those available on your particular system. The MIME type is separated from the application with a semicolon. In many cases, X Window System-based programs are specified. Comments are indicated with a #. A * used in a MIME subtype references all subtypes. The entry **image/*** would be used for an application that can run all types of image files. A formatting code, **%s**, is used to reference the attachment file that will be run on this application. Sample **mailcap** entries are shown here. The first associates all **image** files with the xv image viewer. The next two associate video and video MPEG files with the xanim application.

```
image/*; xv %s
video/*; xanim %s
video/mpeg; xanim %s
```

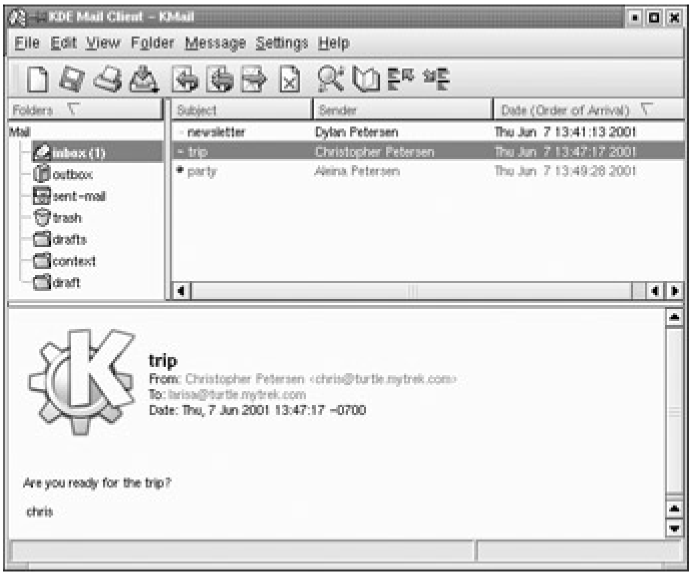
You can also create and edit MIME types on the Gnome and KDE desktops. For Gnome, use the Gnome Control Center's MIME types capplet. This capplet will list the MIME types defined for your system along with their associated filename extensions. Edit an entry to change the application and icon associated with that MIME type, that type of file. On KDE, use the KDE Control Center's File Association entry under File Browsing. This will list MIME types and their associated filename extensions. Select an entry to edit it and change the applications associated with it. KDE saves its MIME type information in a separate file called **mimelnk** in the KDE configuration directory.

Though you can create your own MIME types, a standard set already is in use. The types text, image, audio, video, application, multipart, and message, along with their subtypes, have already been defined for your system. You will find that commonly used file extensions such as .**tif** and .**jpg** for TIFF and JPEG image files are already associated with a MIME type and an application. Though you can easily change the associated application, it is best to keep the MIME types already installed. The current official MIME types are listed at the IANA Web site (**www.iana.org**) under the name Media Types, provided as part of their Assignment Services. You can access the media-types file directly on their FTP site at:

```
ftp.iana.org/in-notes/iana/assignments/media-types/
```

The K Desktop Mail Client: KMail

The K Desktop mail client, KMail, provides a full-featured GUI interface for composing, sending, and receiving mail messages. The KMail window displays three panes for folders, headers, and messages, as shown in Figure 17-1. The upper-left pane displays your mail folders. You have an inbox folder for received mail, an outbox folder for mail you have composed but have not sent yet, and a sent-mail folder for messages you have previously sent. You can create your own mail folders and save selected messages in them, if you wish. The top-right pane displays mail headers for the currently selected mail folder. You can use the scroll bar to the right to move through the list of headers. The headers are segmented according to fields, beginning with sender and subject. A color code is used to indicate read and unread messages. New messages are listed in red. Read messages are in green. A bullet symbol also appears at the beginning of unread message headers. To display a message, click its header. The message is then displayed in the large pane below the header list. You can also send and receive attachments, including binary files. Pictures and movies that are received are displayed using the appropriate K Desktop utility. If you right-click the message, a pop-up menu displays options for actions you may want to perform on it. You can move or copy it to another folder, or simply delete it. You can also compose a reply or forward the message. For secure access, Kmail now supports SSL, provided OpenSSL is installed. It also supports IMAP in addition to POP and SMTP protocols.



Subject	Sender	Date (Order of Arrival)
newsletter	Dylan Petersen	Thu Jun 7 13:41:13 2001
trip	Christopher Petersen	Thu Jun 7 13:47:17 2001
● party	Aleina Petersen	Thu Jun 7 13:49:28 2001

trip
From: Christopher Petersen <chris@turtle.mytrek.com>
To: larisa@turtle.mytrek.com
Date: Thu, 7 Jun 2001 13:47:17 -0700

Are you ready for the trip?

chris

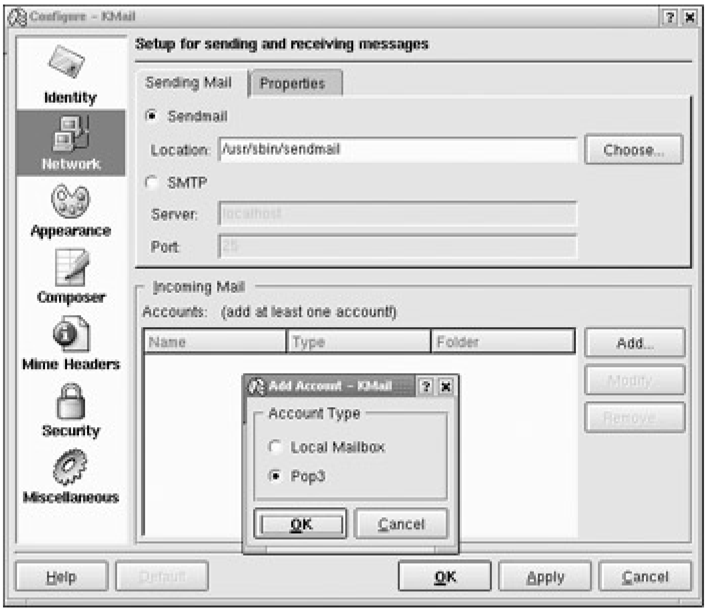
Figure 17-1: The K Desktop mail program

The menus in the menu bar at the top of the window contain the commands and options you can use for managing your mail. An icon bar for commonly used mail commands is displayed below the menu bar. To get new mail, click the icon showing a page with a question mark (?). To print a message, click the Printer icon. To save a message, click the Disk icon. To check your mail, click the Check Mail icon. If you hold the mouse over an icon, a short description of its function is displayed. The icon of an open book opens the KMail address book. Here, you can enter a list of e-mail addresses. Also, right-clicking a displayed message gives you the option of automatically adding its e-mail address to your address book. You can use the Help button or the Help menu to obtain more detailed descriptions of the different KMail features. The Help button opens the KMail Handbook, which provides easy reference to different operations.

You click the Blank Page icon to compose and send a new message. If you want to compose a reply, select the header of the message you want to reply to, and then click the icon showing a page with a single curved arrow. A message window opens with the To entry already filled with the sender's address, the From entry with your address, and the Subject line with the sender's subject with a preceding RE:. To forward a message, select the message's header and click the icon showing a page with two curved arrows.

When you compose a message, a new window opens with entries for the e-mail address, Carbon-copy (CC), and Subject. A button with three dots is placed next to address entries like From and CC. Clicking one of these buttons invokes your K address book, from which you can select an e-mail address. You enter the message in the body of the window. You can use any of the standard mouse-based editing capabilities to cut, copy, paste, and select text. All commands available to you for composing messages are listed in the menus in the menu bar at the top of the window. A button bar of commonly used functions is displayed just below the menu bar. To send the message, click the Envelope button. To attach a file, you can select the Attach entry in the Attach menu or click the Paper Clip button. This menu also has entries for inserting the text of files into the message or appending a signature file. Other composition features, such as the spell-checker and encryption, are also supported. The standard message window does not display all the header entries unless you select All from the View menu. You can also individually select the fields you want displayed. The Options menu enables you to mark a message as urgent or request a delivery confirmation.

To set up KMail for use with your mail accounts, you must enter account information. Select the Configure entry in the Settings menu. Several panels are available on the Settings window, which you can display by clicking their icons in the left column (see Figure 17-2). For accounts, you select the Network panel. Two sections are on this panel: one for sending mail and one for receiving mail. In the sending-mail section, enter the SMTP server you use. If you have an ISP or you are on a LAN, enter the server name for your network. The default is the Sendmail utility on your own Linux system. In the receiving mail section, you can add any mail accounts you may have. You may have more than one mail account on mail servers maintained by your ISP or LAN. A configure window is displayed where you can enter login, password, and host information. The *host* is the name of the POP or IMAP server this particular account uses.



Setup for sending and receiving messages

● Sendmail

Location: /usr/sbin/sendmail

○ SMTP

Server: localhost

Port: 25

Incoming Mail

Accounts: (add at least one account!)

Name	Type	Folder

Add Account – KMail

Account Type

○ Local Mailbox

● Pop3

OK Cancel

Figure 17-2: Kmail configuration

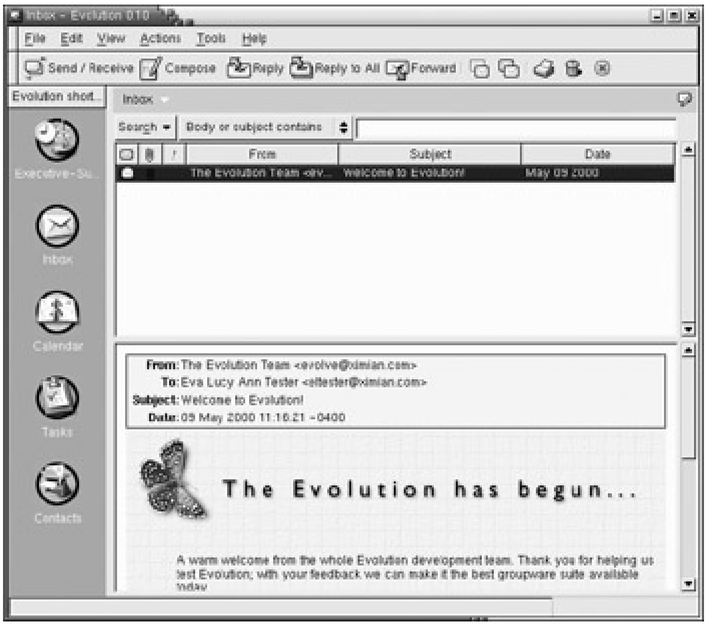
Gnome Mail Clients: Balsa, Evolution, Gmail, Mahogany, and Others

Currently, several Gnome-based mail clients are under development, many of which you can currently use (see Table 17-2). These include Balsa, Mahogany, Gmail, GnoMail, Eucalyptus, Glacier, Pygmy, Spruce, and N-tool (Balsa, Evolution, and Mahogany are discussed further). Check the Gnome Web site for more mail clients as they come out. Many are based on the Gnome mail client libraries (camel) currently under development, which provides support for standard mail operations. Balsa is a Gnome mail client with extensive features, though it can operate under any window manager, including KDE, as long as Gnome is installed on your system. Evolution is an integrated mail client, calendar, and contact manager from Ximian. The Mahogany mail client is a Gnome mail client that also has versions for other platforms. Gmail is meant to be a light and fast e-mail client, supporting basic mail operations, such as forwarding, replies, and mailboxes. GnoMail is yet another Gnome mail client that also uses the Gnome mailer libraries. Glacier is a GNU mail client with extensive MIME support. Eucalyptus is a MIME-compliant mail client that supports unlimited folders, addresses, filters, and multiple POP servers. Eucalyptus was previously implemented on the Amiga. Pygmy is a simple GNOME mail client written in the Python programming language that supports attachments and MIME message. The N-tool is a Gnome mail client with Japanese language support, providing standard features, including mailboxes and full MIME support. Spruce is a Gnome e-mail client with support for multiple accounts. LinPopUp is a port of WinPopUp that operates on Samba-connected networks and can send messages to users on Windows machines running WinPopUp.

Evolution

Evolution is an integrated mail client, calendar, and address book, currently being developed by Ximian. The Evolution mailer is a powerful tool with support for numerous protocols (SMTP, POP, and IMAP), multiple mail accounts, and encryption. With Evolution, you can create multiple mail accounts on different servers, including those that use different protocols such as POP or IMAP. You can also decrypt PGP or GPG-encrypted messages.

The Evolution mailer provides a simple GUI interface, with a toolbar for commonly used commands and a sidebar for shortcuts (see Figure 17-3). A menu of Evolution commands allows access to other operations. The main panel is divided into two panes, one for listing the mail headers and the other for displaying the currently selected message. You can click any header title to sort your headers by that category. Evolution also supports the use of virtual folders. These are folders created by the user to hold mail that meets specified criteria. Incoming mail can be automatically distributed to their particular virtual folder.



			From	Subject	Date
			The Evolution Team <ev...	Welcome to Evolution!	May 09 2000

From: The Evolution Team <evolve@ximian.com>
To: Eva Lucy Ann Tester <eltester@ximian.com>
Subject: Welcome to Evolution!
Date: 09 May 2000 11:16:21 -0400

The Evolution has begun...

A warm welcome from the whole Evolution development team. Thank you for helping us test Evolution; with your feedback we can make it the best groupware suite available today

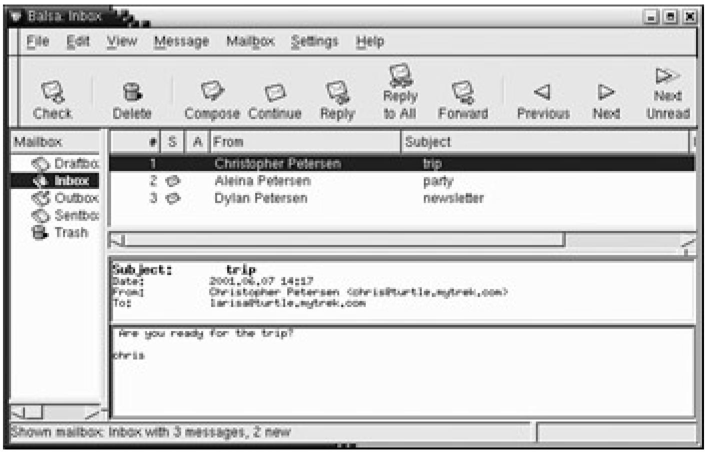
Figure 17-3: Evolution

To create a message, click the Compose icon. This opens a window divided into two sections, one for header information and one for inputting the message. For the headers, you can enter your address, subject, and copy information. You can also use the Evolution address book to automatically enter addresses. You can then use standard GUI editing methods to enter your text. For your message, you can include standard text, images, and HTML (Web page) data. Plug-in Bonobo components will allow you to also include complex data such as audio, video, and PDF data. All such data will be displayed within the message window.

To attach files, click the Attach button on the toolbar. The file selected will show up in a separate pane at the bottom of the window. When you are ready to send your message, click the Send button.

Balsa

Balsa provides a full-featured GUI interface for composing, sending, and receiving mail messages. The Balsa window displays three panes for folders, headers, and messages, as shown in Figure 17-4. The left-side pane displays your mail folders. You initially have three folders: an inbox folder for received mail, an outbox folder for mail you have composed but have not sent yet, and a trash folder for messages you have deleted. You can also create your own mail folders in which you can store particular messages. To place a message in a folder you have created, click and drag the message header for that message to the folder.



#	S	A	From	Subject
1			Christopher Petersen	trip
2			Aleina Petersen	party
3			Dylan Petersen	newsletter

```
Subject:        trip
Date:           2001.06.07 14:17
From:           Christopher Petersen <chris@turtle.mytrek.com>
To:             larisa@turtle.mytrek.com
```

```
Are you ready for the trip?

chris
```

Shown mailbox: Inbox with 3 messages, 2 new

Figure 17-4: Balsa

The right side of the Balsa window consists of two panes. The top-right pane lists the message headers for the currently selected folder. Message headers are displayed showing the subject, sender, and date. An Envelope icon indicates an unread message, and a Trash Can icon indicates a message to be deleted. Headers are segmented into fields with buttons for the fields shown at the top of the pane. You can click these buttons to sort headers by different fields, such as subject or sender. To display a message, you click it. It is then displayed in the pane below the message headers. You can click the Right and Left Arrow icons in the icon bar to move through the header list.

To display message headers for a particular folder, you first must open that folder. Double-clicking the folder's icon both opens the folder and selects it, with its headers displayed. You can also single-click the folder's icon and select Open from the Mailbox menu. This opens the folder and displays a button for it in the bar separating the folder pane from the right-side panes. You can open several folders at once. Each will have its own button in the bar. To have an open folder's message headers display, you can click its button in this middle bar. To close a folder, select it and then choose the Close entry on the Mailbox menu.

You can access commands for managing your mail through the menus on the menu bar located at the top of the Balsa window. An icon bar for commonly used mail commands is displayed below the menu bar. To retrieve new messages, click the Check icon or select the Get New Mail item in the File menu. To print a message, click its header and then the Printer icon. To delete a message, select its header and click the Delete icon. The icons featuring envelopes are different forms of message composition: one for new messages, one for replies, and one for forwarding messages. Balsa also supports filters for automatically performing operations on received mail. You can create a filter that matches a specified string in a header field and then automatically perform an operation on that message. For example, you could have any message from a certain person automatically deleted or messages on a certain subject automatically printed.

To compose a message, click the Compose icon or select the New item on the Message menu. A new message window opens with entries for the To, From, Subject, and Carbon copy header fields. If you select the Reply or Forward icons, the To, From, and Subject fields are already filled in with your address and the sender's address, and the sender's subject with a preceding Re: for replies and Fwd: for forwards. Fields that use addresses have a small Address Book button at the end of their fields. You can click this button to use the address book to enter an address for a field. Enter the message in the body of the window. The standard GUI editing operations are supported, enabling you to use your mouse to select, cut, copy, and move text. To send the message, click the Send icon in the icon bar or select the Send entry in the File menu. Icons also exist for operations such as selecting attachments or printing the message. Attachments added to a message are displayed in a pane just below the icon bar.

You can configure Balsa to access any number of mail accounts. Select the Preferences entry to bring up a window with panels for configuring Balsa. The Mail Servers panel shows three sections: one for remote mailbox servers, another for local mail, and the last for outgoing mail. In the remote mailbox servers section, you can add the server information for your network. Clicking the Add button opens a mailbox configurator window where you can enter your account's username and password, as well as the server name for that mailbox server. The mailbox name is any name you want to use to identify this mail service.

The Mahogany mail client window uses a format similar to Balsa. Three panes are there: a left pane for listing folders, and two other panes on the right side for headers and message text. The headers pane has buttons for sorting headers by different fields, such as subject or sender. Mail operations can be performed using the menus or the button bar at the top of the window. To compose a message, you can click the Envelope button. This opens a window with entries for From, To, and Subject header fields. Mouse-based editing operations, such as cut and paste, are currently not supported, though you can invoke an external editor. A menu bar and an icon bar at the top of the window list the different message operations you can perform, such as the spell-checking and printing.

X Window Mail Clients: Mozilla, Netscape, and exmh

Although many of the newer mail clients are being designed for either Gnome or the K Desktop, several mail clients were developed for use on the X Window System and operate under any window manager or desktop. They do not require either Gnome or the K Desktop. Netscape Messenger, Mozilla mail, and exmh are three of the more popular mail clients. The Emacs mail clients are integrated into the Emacs environment, of which the Emacs editor is the primary application. They are, however, fully functional mail clients. The GNU Emacs mail client can operate either with X Windows capabilities or with a screen-based interface like Pine. The XEmacs mail client operates solely as an X Windows application.

Mozilla Mail and Netscape Messenger

Mozilla is an open source version of Netscape based on Netscape 5.0. It will eventually replace Netscape as the primary Web browser on Red Hat distributions. To use the Mozilla mail client, you have simply select it in the Tasks menu of the Mozilla Web browser or from the Internet menu on the Gnome program menu. When you first start Mozilla, you are prompted to enter new account information. You can add and edit accounts later by selecting the Mail/News Account Settings entry in the Edit menu. This opens a dialog with a button for adding new accounts if you wish.

Mozilla mail will display a window with two panes and a sidebar (see Figure 17-5). The upper one lists headers of received messages and the lower one displays messages. The header information includes the sender, subject, date, priority, and size. Buttons for each across the top of the pane can be used to sort the headers by those categories. To display a message, click its header. The sidebar displays entries for different mail and newsgroup accounts you have set up. You can click the one you want to check. The icon bar displays icons for several common mail operations, such as getting, creating, deleting, or forwarding messages. Click on the GetMail icon to retrieve mail from your mail server.

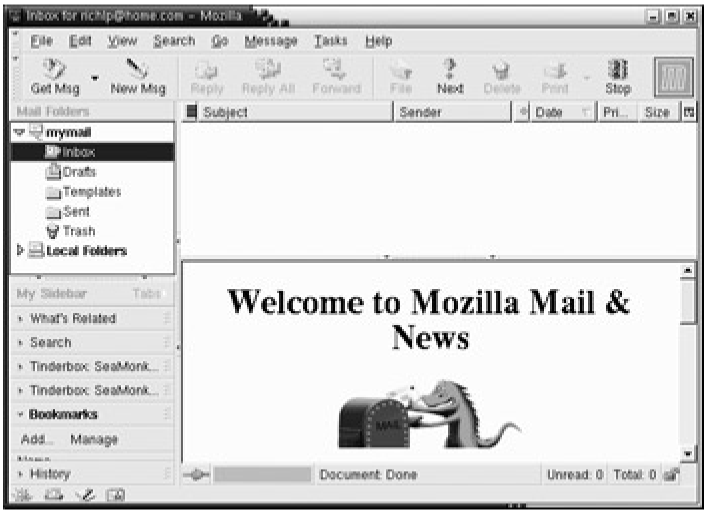


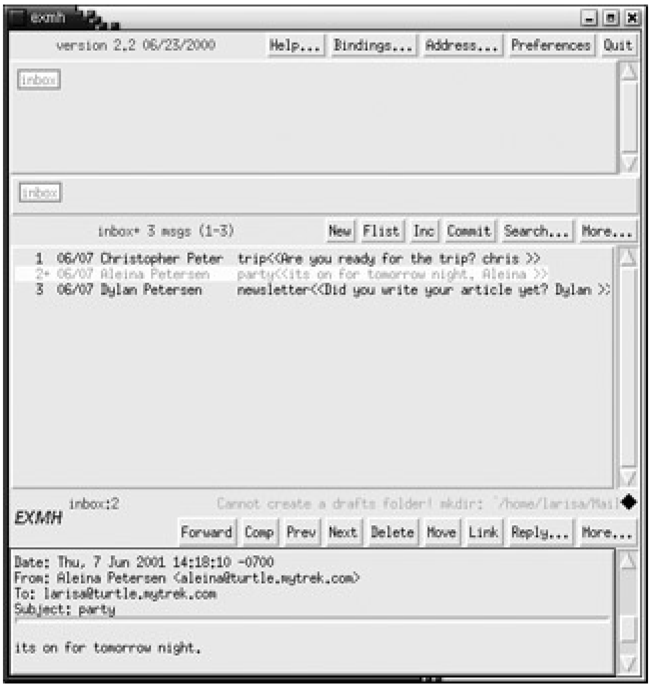
Figure 17-5: Mozilla Mail

To send a message, click the New Msg icon. This opens a window that displays two basic sections. The top section displays entry boxes for header information and attachments. The address box features a drop-down menu for selecting which address field you want to fill. Entries exist for the To, Cc, Bcc, Reply To, Newsgroup, and Followups fields. The attachments box to the right lists files attached to this message. Use the Attachments icon in the icon bar to add attachments. For the text section, an editing toolbar provide icons for a wide range of formatting tasks such as selecting fonts, changing sizes, editing lists, and selecting text colors. Mozilla supports standard GUI editing operations including cut and paste. There are minimize bars on the left of the header, toolbar, editing toolbar, and menu that allow you to minimize them. When you are finished composing your message, click on the Send icon in the toolbar to send it.

Netscape Communicator includes a mail client called Messenger. To use the mail client, you have to select the mail window item in Navigator's window menu or select the Messenger icon in the Communicator window. Account information, such as your mail server, username, and password, must be entered in the Mail panel in the Preferences window, accessible from the Edit menu. Received messages are displayed in the Messenger window. The window is divided into two panes, the upper one listing headers of received messages and the lower one for displaying messages. To display a message, click its header. The icon bar displays icons for several common mail operations, such as sending, deleting, or forwarding messages. A sidebar will list your mail and newsgroup accounts, letting you choose among them. To compose a message, click the New Message icon. Messenger supports a wide range of composition features, such HTML addresses, fonts, formatting, and spell-checker. It supports standard GUI editing operations including cut and paste, though you use the ALT key instead of the CTRL key for keyboard equivalents.

exmh

The exmh program is an X Windows version of the nmh mail client, described later in this chapter. It displays a window with two panes (see Figure 17-6). The upper pane lists the headers for received mail, and the lower pane displays a selected message. Above each pane is a button bar for various nmh commands (these are the same as the commands for the nmh mail client). To check for new mail, you press the Inc button on the top pane. Headers for unread messages are colored blue, and the selected header is displayed in red. You can add new mailbox folders by pressing the New button.



```
1  06/07 Christopher Peter  trip<<Are you ready for the trip? chris >>
2+ 06/07 Aleina Petersen    party<<its on for tomorrow night, Aleina >>
3  06/07 Dylan Petersen     newsletter<<Did you write your article yet? Dylan >>
```

```
Cannot create a drafts folder! mkdir: `/home/larisa/Mail
```

```
Date: Thu, 7 Jun 2001 14:18:10 -0700
From: Aleina Petersen <aleina@turtle.mytrek.com>
To: larisa@turtle.mytrek.com
Subject: party

its on for tomorrow night.
```

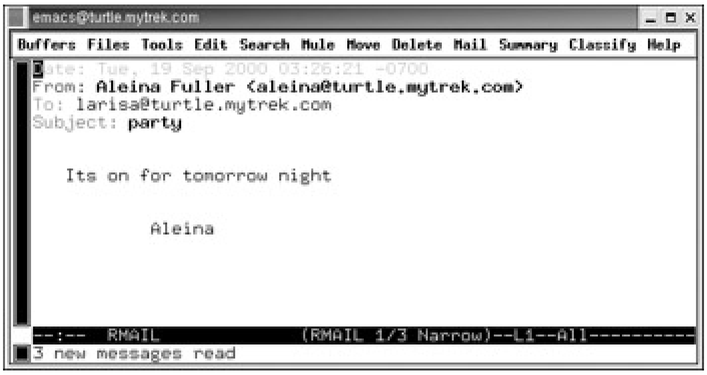
Figure 17-6: exmh

To read a message, click its header, which is displayed in the lower pane. Buttons for managing a message are listed across the top of that pane. Long messages are displayed screen by screen, and you can see the next screen by clicking the More button. The Next and Previous buttons move you directly to the next or previous message. Comp, Reply, and Forward all open a new message window for composing and sending a message. Comp is for new messages, and Reply and Forward include your address, the sender's, and the current message's subject.

To compose a new message, click the Comp button in the lower pane. Header fields and their titles are listed from the top. Click next to a header title and enter its value. For example, to enter a value for the To field, click after "To" and type the address you want. A line below the header separates the header from the text of the message. Click below this line and enter your message. You can change any of the header fields and the text of your message at any time. Click the Send button to send the message.

The Emacs Mail Client: GNU Emacs and XEmacs

The GNU version of Emacs includes a mail client along with other components, such as a newsreader and editor. GNU Emacs is included on Red Hat distributions. Check the Emacs Web site at www.emacs.org for more information. When you start up GNU Emacs, menu buttons are displayed across the top of the screen. If you are running Emacs in an X Windows environment, you have full GUI capabilities and can select menus using your mouse. To access the Emacs mail client, select from the mail entries in the Tools menu. To compose and send messages, just select the Send Mail item in the Tools menu. This opens a screen with prompts for To: and Subject: header entries (see Figure 17-7). You then type the message below them, using any of the Emacs editing capabilities. On the menu bar, a new menu is added labeled Mail. When you are ready to send the mail, choose the Send Mail entry in this menu. To read mail, select the Read Mail item in the Tools menu, which displays the first mail message received. Use entries in the Move menu to move to the next message or back to a previous one, and use entries in the Delete menu to remove a message. The Mail menu lists entries for message operations, such as sending replies or forwarding the message. GNU Emacs is a working environment within which you can perform a variety of tasks, with each task having its own buffer. When you read mail, a buffer is opened to hold the header list, and when you read a message, another buffer will hold the contents. When you compose a message, yet another buffer holds the text you wrote. The buffers you have opened for mail, news, or editing notes or files are listed in the Buffers menu. You can use this menu to switch among them.


emacs@turtle.mytrek.com
Buffers Files Tools Edit Search Mule Move Delete Mail Summary Classify Help

```
Date: Tue, 19 Sep 2000 03:26:21 -0700
From: Aleina Fuller <aleina@turtle.mytrek.com>
To: larisa@turtle.mytrek.com
Subject: party


   Its on for tomorrow night


           Aleina



--:--  RMAIL          (RMAIL 1/3 Narrow)--L1--All----------
3 new messages read
```

Figure 17-7: Emacs mail client

XEmacs is another version of Emacs designed to operate solely with a GUI interface. The Internet applications, which you can easily access from the main XEmacs button bar, include a Web browser, a mail utility, and a newsreader. Clicking the Mail button brings up another window that lists received messages and also enables you to compose new ones. You can compose, reply, or print messages using buttons on the side of the window. To display a message, click its header and press the SPACEBAR. You can display the headers by choosing the Display item in the Folder menu. When composing a message, you have full use of the Emacs editor with all its features, including the spell-checker and search/replace. A new window is opened that prompts you for the address and subject. When you finish editing your message, choose Send and Exit in the Mail menu located at the end of the menu bar.

Screen-Based Mail Clients

You can invoke several powerful mail clients on the command line that provide a full-screen, cursor-based interface. Menus are displayed on the screen whose entries you can select using your keyboard. Basic cursor movement is supported with arrow keys. Pine and Mutt are both mail clients that provide a screen-based interface. Although screen-based, the mail clients are very powerful. Pine, in particular, has an extensive set of features and options.

Pine

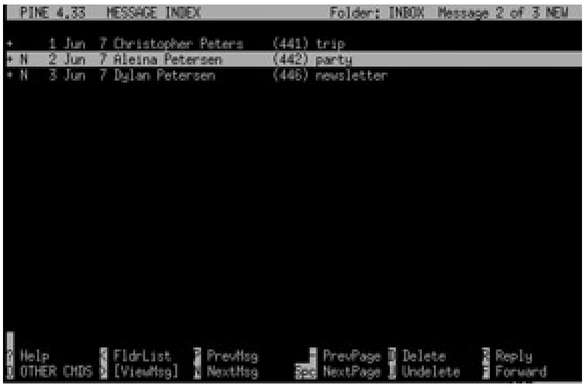
Pine stands for "Program for Internet News and Email." It features full MIME support, enabling you to send messages, documents, and pictures easily. Pine has an extensive list of options, and it has flexible Internet connection capabilities, letting you receive both mail and

Usenet news. Pine also enables you to maintain an address book where you can place frequently used e-mail addresses. You can find more information about Pine, including documentation and recent versions, from the Pine Information Center Web site at **www.washington.edu/pine**. The Pine newsgroup is **comp.mail.pine**, where you can post questions.

Pine runs from the command line using a simple cursor-based interface. Enter the **pine** command to start Pine. Pine supports full-screen cursor controls. It displays a menu whose items you can select by moving the cursor with the arrow keys to the entry of your choice and pressing ENTER. Each item is labeled with a capital letter, which you use to select it. The **O** command brings up a list of other Pine commands you can use.

To send a message, select the Compose Message item. This brings up a screen where you can enter your message. You are first taken through the different entries for the header, which prompts you for an e-mail address and subject. You can even attach files. Then, you type in the text of the message. A set of commands listed at the bottom of the screen specifies different tasks. You can read a file with CTRL-R and cancel the message with CTRL-C. Use CTRL-X to send the message.

Pine organizes both sent and received messages into folders that you select using the Folder List entry on the main menu. The different available folders are listed from left to right. Three folders are automatically set up for you: INBOX, sent-mail, and saved-messages. The INBOX folder holds mail you have received but have not yet read. Sent mail is for messages you have sent to others, while saved messages are messages you have read and want to keep. Use the LEFT and RIGHT ARROW keys to select the one you want and then press ENTER. Selecting the INBOX folder will list the messages you have received, as shown in Figure 17-8. Headers for received messages are then displayed, and you can choose a specific header to view your message. The folder you select becomes your default folder. You can return to it by selecting the Folder Index entry in the main menu.



```
  PINE 4.33   MESSAGE INDEX                    Folder: INBOX  Message 2 of 3 NEW

 +      1 Jun  7 Christopher Peters    (441) trip
 + N    2 Jun  7 Aleina Petersen       (442) party
 + N    3 Jun  7 Dylan Petersen        (446) newsletter





   Help          FldrList    PrevMsg         PrevPage   Delete      Reply
   OTHER CMDS    [ViewMsg]   NextMsg     Spc NextPage   Undelete    Forward
```

Figure 17-8: Selecting messages in Pine

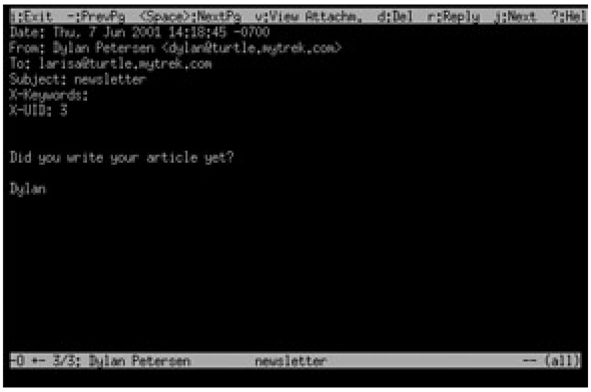
Mutt

Mutt incorporates many of the features of both Elm and Pine. It has an easy-to-use screen-based interface similar to Elm. Like Pine, Mutt has an extensive set of features, such as MIME support. You can find more information about Mutt from the Mutt Web page at **www.mutt.org**. Here, you can download recent versions of Mutt and access online manuals and help resources. On most distributions, the Mutt manual is located in the **/usr/doc** directory under Mutt. The Mutt newsgroup is **comp.mail.mutt**, where you can post queries and discuss recent Mutt developments.

Mutt screens have both an index mode and a pager mode. The *index mode* is used to display and manage message header lists, while the *pager mode* is used to display and compose messages. Mutt screens support ANSI escape sequences for color coding, displaying commands, prompts, and selected entries in different colors. You invoke Mutt with the command **mutt** entered on a Linux shell command line. Mutt displays a list of common commands across the top of the screen. Pressing the single key listed before the command executes that command. For example, pressing **q** quits Mutt, **s** saves the current message, and **r** enables you to send a reply to a message. Press the **?** key to obtain a complete listing of Mutt commands.

To compose a new message, press **m**. On the bottom line, you are then sequentially prompted to enter the address of the person to whom you are sending a message, the subject line, and a carbon copy list. Then you are placed in the standard editor, usually Vi or Emacs, and you can use the editor to enter your message. If you are using Vi, you first have to press the **a** or **i** command before you can enter text (see Chapter 16). After entering your text, you press ESC to return to the Vi command mode. When you finish entering your message, you save and exit Vi with the **ZZ** command. After editing the message, Mutt displays the header and a list of possible commands at the top of the screen. You can then edit the message or any of the header fields again. With the **a** command, you can add attachments to the message, and with the **q** command you can cancel the message. Press the **y** command to send the message.

Headers listing received messages are shown in the main screen upon starting Mutt. You can use the arrow keys to move from one to the next. The selected header is highlighted. Press the ENTER key to display the contents of the message, as shown in Figure 17-9. This opens another screen showing the header fields and the text of the message. Long messages are displayed screen by screen. You can use the PAGE UP or SPACEBAR keys to move to the next screen, and the PAGE DOWN or - keys to move back to the previous screen. The commands for operations you can perform on the message are listed across the top of the screen. With the **r** command, you can compose and send a reply to the message, while the **d** command deletes the message. Once you examine your message, you can use the **i** command to return to the main screen.



```
i:Exit  -:PrevPg  <Space>:NextPg  v:View Attachm,  d:Del  r:Reply  j:Next  ?:Hel
Date: Thu, 7 Jun 2001 14:18:45 -0700
From: Dylan Petersen <dylan@turtle.mytrek.com>
To: larisa@turtle.mytrek.com
Subject: newsletter
X-Keywords:
X-UID: 3


Did you write your article yet?

Dylan















-O +- 3/3: Dylan Petersen          newsletter                                -- (all)
```

Figure 17-9: Reading Mutt messages

Note Another favorite mail client is Elm. Elm has a screen-oriented, user-friendly interface that makes mail tasks easy to execute. However, Elm has been deprecated in Red Hat 7.1 and may be dropped from future releases.

Command Line Mail Clients

Several mail clients use a simple command line interface. They can be run without any other kind of support, such as X Windows, desktops, or cursor support. They are simple and easy to use but include an extensive set of features and options. Two of the more widely used mail clients of this type are Mail and Mail Handler (nmh). Mail is the mailx mail client that was developed for the Unix system. It is considered a kind of default mail client that can be found on all Unix and Linux systems.

Note You can also use the Emacs mail client from the command line, as described in the previous section.

Mail

What is known now as the mail utility was originally created for BSD Unix and called, simply, mail. Later versions of Unix System V adopted the BSD mail utility and renamed it *mailx*. Now, it is simply referred to as *Mail*. Mail functions as a de facto default mail client on Unix and Linux systems. All systems have the mail client called Mail, whereas they may not have other mail clients.

To send a message with Mail, type **mail** along with the address of the person to whom you are sending the message. Press ENTER and you are prompted for a subject. Enter the subject of the message and press ENTER again. At this point, you are placed in input mode. Anything typed in is taken as the contents of the message. Pressing ENTER adds a new line to the text. When you finish typing your message, press CTRL-D on a line of its own to end the message. You will then be prompted to enter a user to whom to send a carbon copy of the message (Cc:). If you do not want to sent a carbon copy, just press ENTER. You will then see *EOT* (*end-of-transmission*) displayed after you press CTRL-D. In the next example, the user sends a message to another user whose address is **robert**. The subject of the message is Birthday. After typing in the text of the message, the user presses CTRL-D, and then presses ENTER to skip the carbon copy prompt.

```
$ mail robert
Subject: Birthday
 Your present is in the mail
 really.

^D
Cc: ENTER

EOT
$
```

The Mail utility receives input from the standard input. By default, the standard input is taken from what the user enters on the keyboard. With redirection, however, you can use the contents of a file as the message for the Mail program. In the next example, the file **mydata** is redirected as input for the Mail utility and sent to **robert**.

```
$ mail robert < mydata
```

You can send a message to several users at the same time by listing those users' addresses as arguments on the command line following the **mail** command. In the next example, the user sends the same message to both **chris** and **aleina**.

```
$ mail chris aleina
```

You may also want to save a copy of the message you are sending for yourself. You can copy a mail message to a file in your account by specifying a filename on the command line after the addresses. The filename must be a relative or full pathname, containing a slash. A pathname identifies an argument as a filename to which Mail saves a copy of the message being sent. In the next example, the user saves a copy of the message to a file called **birthnote**. A relative pathname is used, with the period denoting the current working directory: **./birthnote**.

```
$ mail robert ./birthnote
```

To receive mail, you enter only the **mail** command and press ENTER. This invokes a Mail shell with its own prompt and mail commands. A list of message headers is displayed. Header information is arranged into fields beginning with the status of the message and the message number. The status of a message is indicated by a single uppercase letter, usually **N** for *new* or **U** for *unread.* A message number, used for easy reference to your messages, follows the status field. The next field is the address of the sender, followed by the date and time the message was received, and then the number of lines and characters in the message. The last field contains the subject the sender gave for the message. After the headers, the Mail shell displays its prompt, a question mark, **?**. At the Mail prompt, you enter commands that operate on the messages. The commonly used Mail commands are listed in Table 17-3. An example of a Mail header and a prompt follows:

```
$ mail
Mail version 8.1 6/6/93. Type ? for help.
"/var/spool/mail/larisa": 3 messages 2 unread
 1 chris@turtle.mytrek. Thu Jun 7 14:17 22/554 "trip"
>U 2 aleina@turtle.mytrek Thu Jun 7 14:18 22/525 "party"
 U 3 dylan@turtle.mytrek. Thu Jun 7 14:18 22/528 "newsletter"
& q
```

Table 17-3: Mail Commands

Status Code	**Description**
N	Newly received messages
U	Previously unread messages
R	Reads messages in the current session
P	Preserved messages, read in the previous session and kept in incoming mailbox
D	Deleted messages; messages marked for deletion
O	Old messages
*	Messages you saved to another mailbox file
Display Message	Description
h	Redisplay the message headers
z+ z-	If header list takes up more than one screen, scrolls header list

Table 17-3: Mail Commands

Status Code	**Description**
	forward and backward
t *message-list*	Displays a message referenced by the message list; if no message list is used, the current message is displayed
p *message-list*	Displays a message referenced by the message list; if no message list is used, the current message is displayed
n or +	Displays next message
-	Displays previous message
top *message-list*	Displays the top few lines of a message referenced by the message list; if no message list is used, the current message is displayed
Message List	Description
message-number	References message with message number
num1-num2	References a range of messages beginning with *num1* and ending with *num2*
.	Current message
^	First message
$	Last message
*****	All the messages waiting in the mailbox
/pattern	All messages with *pattern* in the subject field
Address	All messages sent from the user with *address*
:*c*	All messages of the type indicated by *c;* message types are as follows: **n** Newly received messages **o** Old messages previously received **r** Read messages **u** Unread messages **d** Deleted messages
Deleting and Restoring Messages	Description
d *message-list*	Deletes a message referenced by the indicated message list from your mailbox
u *message-list*	Undeletes a message referenced by the indicated message list that has been previously deleted
q	Quits the Mail utility and saves any read messages in the **mbox** file
x	Quits the Mail utility and does *not* erase any messages you deleted; this is equivalent to executing a **u** command on all deleted messages before quitting
pre *message-list*	Preserves messages in your waiting mailbox even if you have already read them

Table 17-3: Mail Commands

Status Code	**Description**
Sending and Editing Messages	Description
r	Sends a reply to all persons who received a message
R	Sends a reply to the person who sent you a message
m *address*	Sends a message to someone while in the Mail utility
v *message-list*	Edits a message with the Vi editor
Saving Messages	Description
s *message-list filename*	Saves a message referenced by the message list in a file, including the header of the message
S *message-list*	Saves a message referenced by the message list in a file named for the sender of the message
w *message-list filename*	Saves a message referenced by the message list in a file without the header; only the text of the message is saved
folder *mailbox-filename*	Switches to another mailbox file
%	Represents the name of incoming mailbox file: **folder %** switches to incoming mailbox file
#	Represents name of previously accessed mailbox file: **folder** # switches to previous mailbox file
&	Represents name of mailbox file used to save your read messages automatically; usually called **mbox**: **folder &** switches to **mbox** file
General Command	Description
?	Displays a list of all the Mail commands
! *command*	Executes a user shell command from within the Mail shell

Mail references messages either through a message list or through the current message marker (>). The greater-than sign (>) is placed before a message considered the current message. The current message is referenced by default when no message number is included with a Mail command. You can also reference messages using a message list consisting of several message numbers. Given the messages in the previous example, you can reference all three messages with **1-3**. The ^ references the first message; for example, **^-3** specifies the range of messages from the first message to the third message. The **$** references the last message. The period, **.**, references the current message. And the asterisk, *****, references all messages. Simply entering the number of the message by itself will display that message. The message is then output screen by screen. Press the SPACEBAR or the ENTER key to continue to the next screen.

You use the **R** and **r** commands to reply to a message you have received. The **R** command entered with a message number generates a header for sending a message and then places you into the input mode to type in the message. The **q** command quits Mail. When you quit, messages you have already read are placed in a file called **mbox** in your home directory. Instead of saving messages in the **mbox** file, you can use the **s** command to save a message explicitly to a file of your choice. The **s** command, however, saves a message with its header, in effect creating another mailbox file. You can then later access a mailbox file either by invoking the Mail utility with the **-f** option and the mailbox filename or, if you are already using Mail, by executing the **folder** command that switches to a specified mailbox file. For example, the command **mail** -f family_msgs accesses the mailbox file **family_msgs**. Each message in the **family_msgs** mailbox file is then displayed in a message list.

Mail has its own initialization file, called **.mailrc**, that is executed each time Mail is invoked, for either sending or receiving messages. Within it, you can define Mail options and create Mail aliases. You can set options that add different features to mail, such as changing the prompt or saving copies of messages you send. To define an alias, you enter the keyword **alias**, followed by the alias you have chosen and then the list of addresses it represents. In the next example, the alias **myclass** is defined in the **.mailrc** file.

.mailrc

```
alias myclass chris dylan aleina justin larisa
```

In the next example, the contents of the file **homework** are sent to all the users whose addresses are aliased by **myclass**.

```
$ mail myclass < homework
```

The New Mail Handler Utility: nmh

The Mail Handler mail client, commonly known as nmh, takes a different approach to managing mail than most other mail clients. nmh consists of a set of commands you execute within your user shell, just as you would execute any other Unix command. No special mail shell exists, as there is for Mail. One nmh command sends a message, another displays your incoming messages, and still another saves a message. The nmh commands and their options are listed in Table 17-4. A set of environment variables provides a context for the nmh commands you execute, such as keeping track of the current messages or mail folders.

Table 17-4: nmh Commands

Command	**Description**
inc	Places received mail in your incoming mailbox and displays message headers
show *num*	Displays current message or specified messages
prev	Displays the previous message
next	Displays the next message
scan	Redisplays message headers
Mhl	Displays formatted listing of messages
folders	Lists all mail folders
forw	Forwards a message
repl	Replies to a message

Table 17-4: nmh Commands

Command	**Description**
send	Resends a message or sends a file as a message
pick	Selects messages by specified criteria and assigns them a sequence
folder	Changes to another mailbox file (folder)

Note Instead of working from a command line interface, you can use xmh or exmh, which provides an X Windows interface for accessing nmh messages.

To send a message using nmh, you first need to compose the message using the **comp** command, and then send the message with the **send** command. To compose a message, type in the word **comp** on the command line by itself and press ENTER. With nmh, you are placed in an input mode for the default editor used for nmh (usually the Vi editor). Fields at the top of the screen show prompts for the address, carbon copy, and subject. Below the dotted line, you enter your message. You can use your arrow keys to move from one field to another. Once you type the contents of the message, save and quit the editor as you normally would (ESC-SHIFT-ZZ for Vi). At the **What now?** prompt, you can send the message, edit it, save it to a file, display it again, or quit without sending the message. The **send** command sends the message. Pressing ENTER at the **What now?** prompt displays a list of commands you can enter. In the next example, the user composes a message for another user whose address is **robert**.

```
$ comp
To: robert
cc:
Subject: Birthday
-------------------
Your present is in the mail
really.

What now? send
$
```

To read your mail with nmh, you first need to store newly received mail into a designated nmh mailbox file with the **inc** command. The **inc** command displays a list of headers for each mail message in your incoming mailbox. An nmh message header consists only of the message number, the month and year, the address of the sender, and the beginning of the message text.

```
 $ inc
 1+ 06/07 Christopher Peter trip<<Are you ready for the trip? chris >>
 2 06/07 Aleina Petersen party<<its on for tomorrow night. Aleina >>
 3 06/07 Dylan Petersen newsletter<<Did you write your article yet? Dyla
 $
```

If you want to redisplay the headers, you need to use another nmh command called **scan**.

```
 $ scan
 1+ 06/07 Christopher Peter trip<<Are you ready chris >>
 2 06/07 Aleina Petersen party<<its on for Aleina >>
 3 06/07 Dylan Petersen newsletter<<Did you Dylan >>
 $
```

You use the **show**, **next**, and **prev** commands to display a message. The **show** command displays the current message, the **next** command displays the message after the current one, and the **prev** command displays the message before the current one. Initially, the current message is the first of the newly received messages. If you want to display a particular message, you can use the **show** command with the number of the message. **show 2** displays message 2. You can also reference several messages at once by listing their message numbers. The command **show 1 3** displays messages 1 and 3. You can also designate a range of messages by specifying the first message number in the range, and the last number, separated by a minus sign. **show 1-3** displays messages 1, 2, and 3.

```
$ show
$ next
```

To print a message, you first output it with **show,** and then pipe the output to a printer. You save a message to a text file in much the same way. First you output the message using the **show** command, and then you redirect that output to a file.

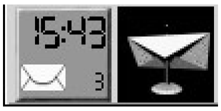
```
$ show | lpr
$ show > myfile
```

You reply to the current message using the **repl** command. You need to know either the message number or the address and subject of the message to reply to it. You delete the current message using the **rmm** command. To delete a specific message, use the message number with **rmm**. rmm 2 deletes the second message. You can create your own mailbox files for nmh using the **folder** command. nmh mailbox files are commonly referred to as *folders.* To create a new folder, enter the **folder** command followed by the name of your folder preceded by a + sign. The + sign identifies an argument as a folder name.

```
$ repl 2
$ rmm 2
$ folder +mybox
```

Notifications of Received Mail

As your mail messages are received, they are automatically placed in your mailbox file, but you are not automatically notified when you receive a message. To find out if you have any messages waiting, either you can use a mail client to retrieve messages or you can use a mail monitor tool to tell you if you have any mail waiting. There are also a number of mail monitors available for use on Gnome. Several operate as applets on the Gnome panel. On the Red Hat Gnome desktop, there are two mail monitors you can choose from: the Mail Check and the Clock and Mail Notify monitors. Both are applets that run inside a Gnome panel. The Mail Check applet will display a mail envelope when mail arrives, and the Clock and Mail Notify applet displays a small envelope and the number of messages received below the time. Both are shown next, with Clock and Mail Notify on the left and Mail Check on the right. Other applets like Sonicmail will notify you of any POP3 mail arrivals. PyBiff performs much the same kind of mail monitoring as Korn. gbox_applet will monitor mailboxes, assigning priorities to each. GMailWatch is a mail monitor applet that will display a summary of incoming mail.



The Red Hat KDE Desktop has a mail monitor utility called Korn that works in much the same way. Korn shows an empty inbox tray when there is no mail and a tray with slanted letters in it when mail arrives. If old mail is still in your mailbox, letters are displayed in a neat square. You can set these icons as any image you want. You can also specify the mail client to use and the polling interval for checking for new mail. If you have several mail accounts, you can set up a Korn profile for each one. Different icons can appear for each account, telling you when mail arrives in one of them.

If you are just using a window manager, such as fvwm2 or Enlightenment, you can use the xbiff utility to perform the same function. xbiff displays an icon of a mailbox, which has a flag on it, on your desktop. When mail arrives, the flag goes up. xbiff can also beep or produce some other sound, if you prefer.

For command line interfaces you can use the biff utility. The biff utility notifies you immediately when a message is received. This is helpful when you are expecting a message and want to know as soon as it arrives. biff automatically displays the header and beginning lines of messages as they are received. To turn on biff, you enter **biff y** on the command line. To turn it off, you enter **biff n**. To find out if biff is turned on, enter **biff** alone.

You can temporarily block biff by using the **mesg n** command to prevent any message displays on your screen. **mesg n** not only stops any Write and Talk messages, it also stops biff and Notify messages. Later, you can unblock biff with a **mesg y** command. A **mesg n** command comes in handy if you don't want to be disturbed while working on some project.

Accessing Mail on Remote POP Mail Servers

Most newer mail clients are equipped to access mail accounts on remote servers. For such mail clients, you can specify a separate mail account with its own mailbox. For example, if you are using an ISP, most likely you will use that ISP's mail server to receive mail. You will have set up a mail account with a username and password for accessing your mail. Your e-mail address is usually your username and the ISP's domain name. For example, a username of **larisa** for an ISP domain named **mynet.com** would have the address **larisa@mynet.com**. The username would be **larisa**. The address of the actual mail server could be something like **mail.mynet.com**. The user **larisa** would log into the **mail.mynet.com** server using the username **larisa** and password to access mail sent to the address **larisa@mynet.com**. Newer mail clients, such as KMail, Balsa, and Netscape, enable you to set up a mailbox for such an account and access your ISP's mail server to check for and download received mail. You must specify what protocol a mail server uses. This is usually the Post Office Protocol (POP). This procedure is used for any remote mail server. Using a mail server address, you can access your account with your username and password.

Instead of creating separate mailboxes in different mail clients, you can arrange to have mail from different accounts sent directly to the inbox maintained by your Linux system for your Linux account. All your mail, whether from other users on your Linux system or from ISP mail servers, will appear in your local inbox. Such a feature is helpful if you are using a mail client, such as Elm or Mail, that does not have the capability to access mail on your ISP's mail server. You can implement such a feature with Fetchmail. Fetchmail checks for mail on remote mail servers and downloads it to your local inbox, where it appears as newly received mail.

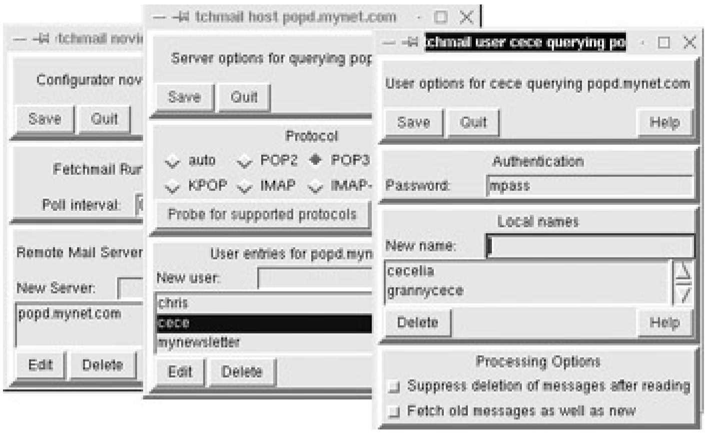
To use Fetchmail, you have to know a remote mail server's Internet address and mail protocol. Most remote mail servers use the POP3 protocol, but others may use the IMAP, ETRM, or POP2 protocols. Enter **fetchmail** on the command line with the mail server address and any needed options. The mail protocol is indicated with the **-p** option and the mail server type, usually POP3. If your e-mail username is different from your Linux login name, you use the **-u** option and the e-mail name. Once you execute the **fetchmail** command, you are prompted for a password. The syntax for the **fetchmail** command for a POP3 mail server follows:

```
fetchmail -p POP3 -u username mail-server
```

To use Fetchmail, connect to your ISP and then enter the **fetchmail** commands with the options and the POP server name on the command line. You will see messages telling you if mail is there and, if so, how many messages are being downloaded. You can then use a mail client to read the messages from your inbox. You can run Fetchmail in daemon mode to have it automatically check for mail. You have to include an option specifying the interval in seconds for checking mail.

```
fetchmail -d 1200
```

You can specify options such as the server type, username, and password in a **.fetchmailrc** file in your home directory. You can also have entries for other mail servers and accounts you may have. Instead of entering options directly into the **.fetchmailrc** file, you can use the fetchmailconf program. fetchmailconf provides a GUI interface for selecting Fetchmail options and entering mail account information. fetchmailconf runs only under X Windows and requires that python and Tk be installed (Red Hat 7.1 does not install fetchmailconf as part of the standard install-you will have to install it manually). It displays windows for adding news servers, configuring a mail server, and configuring a user account on a particular mail server. The expert version displays the same kind of windows, but with many more options. Initially, fetchmailconf displays a window with buttons for choosing a novice or expert version (see Figure 17-10). Choosing the novice version displays a window with an entry labeled "New Server." Type the address of your mail server in the adjoining box and press ENTER. The server address then appears in a list below. To configure that server, click the server name and then the Edit button at the bottom of the window. A new window opens with entries such as user accounts and server protocols. You can add as many user accounts as you may have on that server. You can then further configure an individual account by selecting the username and clicking the Edit button. This opens another window for user account options. You can specify a password and specify any corresponding local users for which you want mail for this account downloaded.



Password: mpass

Protocol: ○ auto ○ POP2 ● POP3 ○ KPOP ○ IMAP ○ IMAP-

popd.mynet.com

chris
cece
mynewsletter

cecelia
grannycece

□ Suppress deletion of messages after reading
□ Fetch old messages as well as new

Figure 17-10: fetchmailconf

Once it is configured, you can enter **fetchmail** with no arguments; it will read entries from your **.fetchmailrc** file. Accounts you have specified are checked, and any new mail is placed in your inbox. If you want Fetchmail to check automatically for new mail periodically, you can activate its daemon mode. To do so, place a daemon entry in the **.fetchmailrc** file. The following entry activates the Fetchmail daemon mode, checking for mail every 1,200 seconds:

```
Set daemon 1200
```

You can also make entries directly in the **.fetchmailrc** file. An entry in the **.fetchmailrc** file for a particular mail account consists of several fields and their values: poll, protocol, username, and password. *Poll* is used to specify the mail server name, and *protocol,* the type of protocol used. Notice you can also specify your password, instead of having to enter it each time Fetchmail accesses the mail server. The syntax for an entry follows:

```
poll SERVERNAME protocol PROTOCOL username NAME password PASSWORD
```

You can use abbreviations for certain field names if you want: *proto* for protocol, *user* for username, and *pass* for password. An example follows for a POP3 server and an account with the username **chris** and the password **mypass**:

```
poll popd.mynet.com proto pop3 user chris password mypass
```

You can specify a default entry for any of these fields and not have to repeat them for each account entry. The default must be placed before the mail server entries. The following example sets the default protocol to POP3 and the username to **chris**:

```
defaults protocol pop3 user chris
```

This next example would reference the **chris** account with the password **newpass** on the **popd.train.com** mail server using the POP3 protocol. The missing fields are filled in by default.

```
poll popd.train.com password newpass
```

Fetchmail enables you to download messages to a specific user on your local system. In fact, you can access several accounts on the remote system and have them downloaded to specific users on the your local system. This is useful when running Fetchmail in daemon mode.

Essentially, Fetchmail is transferring mail from one set of remote accounts to corresponding ones on your local system. You could even have Fetchmail download from one remote account to several local ones, sending copies of the same mail to each. This is helpful if you are using several accounts on your Linux system, or if a group of users is using an account on the remote server for group mail. Local users are specified with the keyword **is** or **to** followed by the usernames, terminating with the keyword **here**. The following examples show different ways of specifying local users. The last entry will send all mail retrieved from the **mynewsletter** account to the users **larisa**, **aleina**, and **dylan**.

```
poll popd.mynet.com proto pop3 user chris password mypass is chris here
poll popd.othernet.com proto pop3 user neil password mypass is chris here
poll popd.mynet.com proto pop3 user cece password mypass to cecelia
grannycece here
poll popd.mynet.com proto pop3 user mynewsletter password mypass to larisa
aleina dylan here
```

Note Fetchmail also supports a multidrop mailbox feature. You can have several users' mail sent to one mailbox on the mail server, and then download it from there to the inboxes for their Linux accounts.

Chapter 18: Usenet and Newsreaders

Overview

Usenet is an open mail system on which users post news and opinions. It operates like a system-wide mailbox that any user on your Linux system can read or send messages to. Users' messages are incorporated into Usenet files, which are distributed to any system signed up to receive them. Each system that receives Usenet files is referred to as a *site.* Certain sites perform organizational and distribution operations for Usenet, receiving messages from other sites and organizing them into Usenet files, which are then broadcast to many other sites. Such sites are called *backbone sites*, and they operate like publishers, receiving articles and organizing them into different groups.

To access Usenet news, you need access to a news server. A news server receives the daily Usenet newsfeeds and makes them accessible to other systems. Your network may have a system that operates as a news server. If you are using an Internet service provider (ISP), a news server is probably maintained for your use. To read Usenet articles, you use a *newsreader*-a client program that connects to a news server and accesses the articles. On the Internet and in TCP/IP networks, news servers communicate with newsreaders using the Network News Transfer Protocol (NNTP) and are often referred to as NNTP news servers. Or, you could also create your own news server on your Linux system to run a local Usenet news service or to download and maintain the full set of Usenet articles. Several Linux programs, called *news transport agents,* can be used to create such a server. This chapter focuses on the variety of news readers available for the Linux platform. The configuration administration and architecture of the NNTP server are covered in Chapter 27.

Usenet News

Usenet files were originally designed to function like journals. Messages contained in the files are referred to as *articles.* A user could write an article, post it in Usenet, and have it immediately distributed to other systems around the world. Someone could then read the article on Usenet, instead of waiting for a journal publication. Usenet files themselves were organized as journal publications. Because journals are designed to address specific groups, Usenet files were organized according to groups called *newsgroups*. When a user posts an article, it is assigned to a specific newsgroup. If another user wants to read that article, he or she looks at the articles in that newsgroup. You can think of each newsgroup as a constantly updated magazine. For example, to read articles on computer science, you would access the Usenet newsgroup on computer science. More recently, Usenet files have also been used as bulletin boards on which people carry on debates. Again, such files are classified into newsgroups, though their articles read more like conversations than journal articles. You can also create articles of your own, which you can then add to a newsgroup for others to read. Adding an article to a newsgroup is called *posting* the article.

Each newsgroup has its own name, which is often segmented to classify newsgroups. Usually, the names are divided into three segments: a general topic, a subtopic, and a specific topic. The segments are delimited by periods. For example, you may have several newsgroups dealing with the general topic *rec,* which stands for recreation. Of those, some newsgroups may deal with only the subtopic food. Again, of those, a group may only discuss a specific topic, such as recipes. In this case, the newsgroup name would be **rec.food.recipes**.

Many of the bulletin board groups are designed for discussion only, lacking any journal-like articles. A good number of these begin with either alt or talk as their general topic. For example, **talk.food.chocolate** may contain conversations about how wonderful or awful chocolate is perceived, while **alt.food.chocolate** may contain informal speculations about the importance of chocolate to the basic structure of civilization as we know it. Here are some examples of Usenet newsgroup names:

```
comp.ai.neural-nets
comp.lang.pascal
sci.physics.fusion
rec.arts.movies
rec.food.recipes
talk.politics.theory
```

Linux has newsgroups on various topics. Some are for discussion, and others are sources of information about recent developments. On some, you can ask for help for specific problems. A selection of some of the popular Linux newsgroups is provided here:

Newsgroup	Topic
comp.os.linux.announce	Announcements of Linux developments
comp.os.linux.admin	System administration questions
comp.os.linux.misc	Special questions and issues
comp.os.linux.setup	Installation problems
comp.os.linux.help	Questions and answers for particular problems
linux.help	Obtain help for Linux problems

Numerous Red Hat-specific newsgroups are also available, some of which are listed here:

Newsgroup	Topic

Newsgroup	Topic
linux.redhat	Red hat Linux topics
linux.redhat.development	Application development issues on Red Hat Linux
linux.redhat.install	Red Hat installation questions
linux.redhat.misc	Miscellaneous questions
linux.redhat.rpm	Discuss problems with the Red Hat Package Manager (RPM)
redhat.config	Red Hat configuration questions
redhat.networking.general	Red Hat networking problems
redhat.security.general	Security issues with Red Hat
redhat.kernel.general	Working with the Red Hat Linux kernel

You read Usenet articles with a newsreader, such as KNode, Pan, Pine, Mozilla, Netscape, trn, or tin, which enables you first to select a specific newsgroup and then read the articles in it. A newsreader operates like a user interface, enabling you to browse through and select available articles for reading, saving, or printing. Most newsreaders employ a sophisticated retrieval feature called *threads* that pulls together articles on the same discussion or topic. Newsreaders are designed to operate using certain kinds of interfaces. For example, KNode is a KDE newsreader that has a KDE interface and is designed for the KDE desktop. Pan has a Gnome interface and is designed to operate on the Gnome desktop. Pine is a cursor-based newsreader, meaning that it provides a full-screen interface that you can work with using a simple screen-based cursor that you can move with arrow keys. It does not support a mouse or any other GUI feature. trn uses a simple command line interface with limited cursor support. Most commands you type in and press ENTER to execute. Several popular newsreaders are listed in Table 18-1.

Table 18-1: Linux Newsreaders

Newsreader	Description
Pan	Gnome Desktop newsreader
KNode	KDE Desktop newsreader
Mozilla	Web utility with newsreader capabilities (X Windows based)
Netscape	Web utility with newsreader capabilities (X Windows based)
Pine	Mail client with newsreader capabilities (cursor based)
Slrn	Newsreader (cursor based)
Emacs	Emacs editor, mail client, and newsreader (cursor based)
trn	Newsreader (command line interface)
tin	Newsreader (command line interface)

Note Numerous newsreaders currently are under development for both Gnome and KDE. You can check for KDE newsreaders on the software list on the K Desktop Web site at **apps.kde.com**. For Gnome newsreaders, check Internet tools on the software map on the Gnome Web site at **www.gnome.org**. The Mozilla newsreader is integrated into the Mozilla Web browser and is available from **www.mozilla.org**.

Most newsreaders can read Usenet news provided on remote news servers that use the NNTP. Many such remote news servers are available through the Internet. Desktop newsreaders, such as KNode and Pan, have you specify the Internet address for the remote news server in their own configuration settings. Several shell-based newsreaders, however, such as trn, tin, and Pine, obtain the news server's Internet address from the **NNTPSERVER** shell variable. Before you can connect to a remote news server with such newsreaders, you first have to assign the Internet address of the news server to the **NNTPSERVER** shell variable, and then export that variable. You can place the assignment and export of **NNTPSERVER** in a login initialization file, such as **.bash_profile**, so it is performed automatically whenever you log in. Administrators could place this entry in the **/etc/profile** file for a news server available to all users on the system.

```
$ NNTPSERVER=news.servdomain.com
$ export NNTPSERVER
```

Chapter 19: FTP Clients

Overview

The Internet is a network of computers around the world you can access with an Internet address and a set of Internet tools. Many computers on the Internet are configured to operate as servers, providing information to anyone who requests it. The information is contained in files you can access and copy. Each server, often referred to as a *site,* has its own Internet address by which it can be located. Linux provides a set of Internet tools you can use to access sites on the Internet, and then locate and download information from them. These tools are known as *clients.* A client application, such as an FTP or a Web client, can communicate with a corresponding server application running on a remote system. An FTP client can communicate with an FTP server program on another system. The server lets the client access certain specified resources on its system and lets an FTP client transfer certain files. Several popular FTP clients are shown in Table 19-1.

Table 19-1: Linux FTP Clients

FTP Client	**Description**
Konquerer	The K Desktop file manager
Nautilus and GNU Midnight Commander	The Gnome file managers
gFTP	Gnome FTP client
NcFTP	Screen based FTP client
ftp	Command line FTP client

To access Internet sites, your computer must be connected to the Internet. You may be part of a network already connected to the Internet. If you have a stand-alone computer, such as a personal computer, you can obtain an Internet connection from an Internet service provider (ISP). Once you have an Internet address of your own, you can configure your Linux system to connect to the Internet and use various Internet tools to access different sites. Chapter 36 describes how to configure your Linux system to make such a connection.

The primary tools for accessing Internet sites are FTP clients and Web browsers. With FTP clients, you can connect to a corresponding FTP site and download files from it. FTP clients are commonly used to download software from FTP sites that operate as software repositories. Most Linux software applications can be downloaded to your Linux system from such sites. A distribution site like **ftp.redhat.com** is an example of one such FTP site, holding an extensive set of packaged Linux applications you can download using an FTP client and then easily install on your system. In the last few years, the Web browsers have become the primary tool for accessing information on the Internet. Most of the tasks you perform on the Internet may be done easily with a Web browser. You only need to use an FTP client to download or upload files from or to a specific FTP site.

Other Internet tools are also available for your use, such as telnet and IRC clients. The *telnet* protocol enables you to log into an account directly on another system. *IRC clients* set up chat rooms through which you can communicate with other users over the Internet. Web clients are discussed in the next chapter, and telnet and IRC clients are discussed in Chapter 21.

Internet Addresses

The Internet uses a set of network protocols called TCP/IP, which stands for Transmission Control Protocol/Internet Protocol. In a TCP/IP network, messages are broken into small components called *datagrams,* which are then transmitted through various interlocking routes and delivered to their destination computers. Once received, the datagrams are reassembled into the original message. Datagrams are also referred to as *packets.* Sending messages as small components has proved far more reliable and faster than sending them as one large bulky transmission. With small components, if one is lost or damaged, only that component has to be resent, whereas if any part of a large transmission is corrupted or lost, the entire message must be resent.

On a TCP/IP network such as the Internet, each computer is given a unique address called an *IP address.* The IP address is used to identify and locate a particular host-a computer connected to the network. It consists of a number, usually four sets of three numbers separated by periods. An example of an IP address is **192.168.187.4.** IP addressing is described in detail in Chapter 39. Non-Internet machines use a gateway to connect to the Internet (see Chapters 25 and 39).

All hosts on the Internet are identified by their IP addresses. When you send a message to a host on the Internet, you must provide its IP address. Using a sequence of four numbers of an IP address, however, can be difficult. They are hard to remember, and it's easy to make mistakes when typing them. To make identifying a computer on the Internet easier, the Domain Name Service (DNS) was implemented. The DNS establishes a domain name address for each IP address. The domain name address is a series of names separated by periods. Whenever you use a domain name address, it is automatically converted to an IP address, which is then used to identify that Internet host. The domain name address is far easier to use than its corresponding IP address.

A domain name address needs to be registered with an Internet domain name registry like the American Registry for Internet Number (ARIN) so that each computer on the Internet can have a unique name (see **www.iana.org** for more information). Creating a name follows specified naming conventions. The domain name address consists of the hostname, the name you gave to your computer; a domain name, the name that identifies your network; and an extension that identifies the type of network you are on. Here is the syntax for domain addresses:

```
host-name.domain-name.extension
```

In the following example, the domain address references a computer called *metalab* on a network referred to as *unc.* It is part of an educational institution, as indicated by the extension *edu.*

```
metalab.unc.edu
```

With the **whois** command, you can obtain information for domain name servers about different networks and hosts connected to the Internet. Enter **whois** and the domain name address of the host or network, and **whois** displays information about the host, such as the street address and phone number, as well as contact persons.

```
$ whois domain-address
```

Network File Transfer: FTP

You can use File Transfer Protocol (FTP) clients to transfer extremely large files directly from one site to another. FTP can handle both text and binary files. This is one of the TCP/IP protocols, and it operates on systems connected to networks that use the TCP/IP protocols, such as the Internet. FTP performs a remote login to another account on another system connected to you on a network, such as the Internet. Once logged into that other system, you can transfer files to and from it. To log in, you need to know the login name and password for the account on the remote system. For example, if you have accounts at two different sites on the Internet, you can use FTP to transfer files from one to the other. Many sites on the Internet allow public access using FTP, however. Many sites serve as depositories for large files anyone can access and download. Such sites are often referred to as *FTP sites,* and in many cases, their Internet address begins with the word *ftp,* such as **ftp.gnome.org** or **ftp.redhat.com**. Others begin with other names, such as **metalab.unc.edu**. These public sites allow anonymous FTP login from any user. For the login name, you use the word "anonymous," and for the password you use your Internet address. You can then transfer files from that site to your own system.

You can perform FTP operations using any one of a number of FTP client programs. For Linux systems, you can choose from several FTP clients. Many now operate using GUI interfaces such as Gnome. Some, such as Netscape, have limited capabilities, whereas others, such as NcFTP, include an extensive set of enhancements. The original FTP client is just as effective, though not as easy to use. It operates using a simple command line interface and requires no GUI or cursor support, as do other clients.

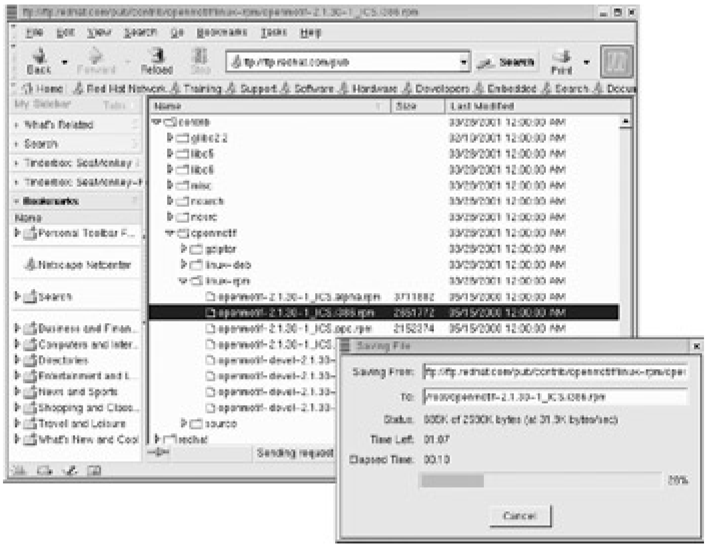
Online FTP Resources

The Internet has a great many sites open to public access. They contain files anyone can obtain using file transfer programs, such as NcFTP. Unless you already know where a file is located, however, finding it can be difficult. To search for files on FTP sites, you can use search engines provided by Web sites, such as Yahoo!, Excite, Alta Vista, Google, or Lycos. For Linux software, you can check sites such as **freshmeat.net**, **sourceforge.net**, rpmfind.net, apps.kde.com, and www.linuxapps.com. These sites usually search for both Web pages and FTP files.

Note Linux tools like Ganesha and Karchie will search FTP sites for requested software.

Web Browser-Based FTP: Netscape

You access an FTP site and download files from it with any Web browser. A Web browser is effective for checking out an FTP site to see what files are listed there. When you access an FTP site with a Web browser, the entire list of files in a directory is listed as a Web page. You can move to a subdirectory by clicking its entry. Click the double periond entry (**..**)at the top of the page to move back up to the parent directory. With Mozilla or Netscape Navigator, you can easily browse through an FTP site to download files. To download a file with Mozilla, hold down the SHIFT key and then double-click the file (for Netscape, hold the SHIFT key down and single-click). This will start the transfer operation. This opens a box for selecting your local directory and the name for the file (see Figure 19-1). The default name is the same as on the remote system. Mozilla and Netscape Navigator have some important limitations. You cannot upload a file, and you cannot download more than one file at a time. They are useful for locating individual files, though not for downloading a large set of files, as is usually required for a system update.



Saving From: ftp://ftp.redhat.com/pub/contrib/openmotif/linux-rpm/ope

To: /root/openmotif-2.1.30-1_ICS.i386.rpm

Status: 800K of 2500K bytes (at 31.3K bytes/sec)

Time Left: 01:07

Elapsed Time: 00:10

33%

Figure 19-1: Mozilla FTP file transfer

The K Desktop File Manager: Konqueror

On the K Desktop, the desktop file manager (Konqueror) has a built-in FTP capability, as shown in Figure 19-2. The FTP operation has been seamlessly integrated into standard desktop file operations. Downloading files from an FTP site is as simple as copying files by dragging them from one directory window to another, but one of the directories happens to be located on a remote FTP site. On the K Desktop, you can use a file manager window to access a remote FTP site. Files in the remote directory are listed just as your local files are. To download files from an FTP site, you open a window to access that site, entering the URL for the FTP site in the window's location box. Open the directory you want, and then open another window for the local directory to which you want the remote files copied. In the window showing the FTP files, select the ones you want to download. Then, simply click and drag those files to the window for the local directory. A pop-up menu appears with choices for Copy, Link, or Move. Select Copy. The selected files are then downloaded. Another window then opens, showing the download progress and displaying the name of each file in turn, and a bar indicating the percentage downloaded so far.

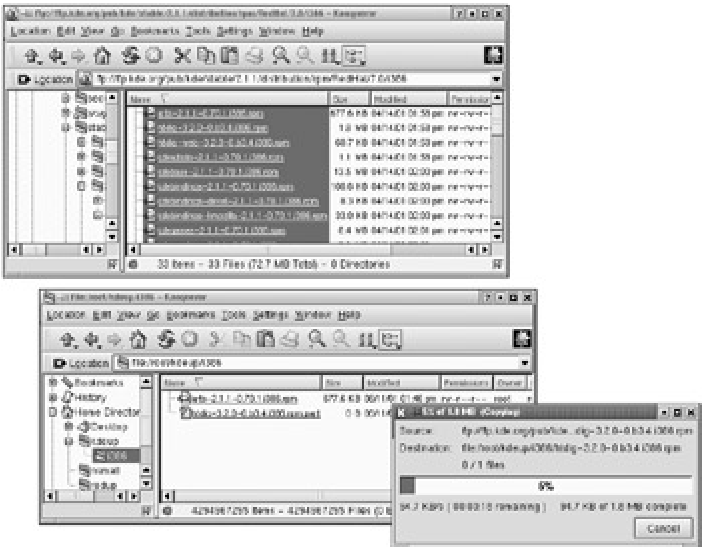


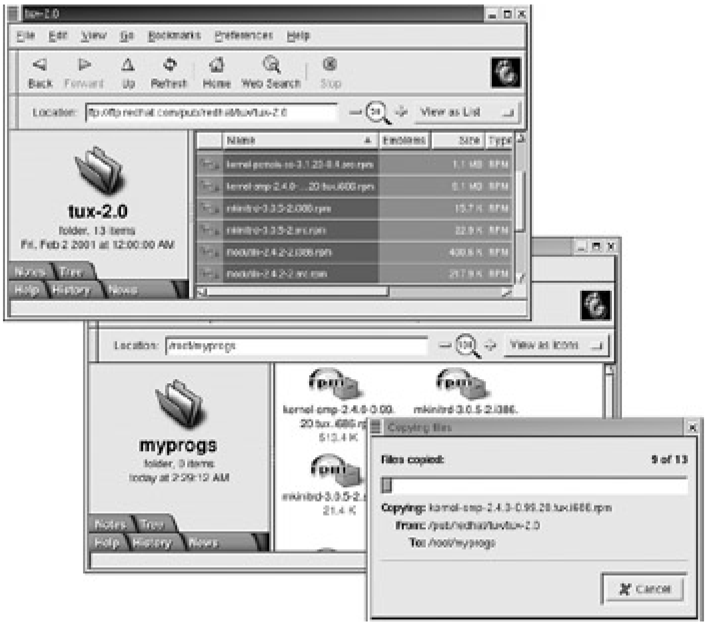
Figure 19-2: The K Desktop file manager (Konqueror) performing FTP operations

Gnome FTP: GNU Midnight Commander, gFTP, Nautilus

The easiest way to download files is to use the built-in FTP capabilities of the Gnome file manager, Midnight Commander. You can also use several Gnome-based FTP clients that offer more features, including gFTP. Check the Gnome Web site at www.gnome.org for more. gFTP is included with the current Gnome release.

Gnome File Manager

On Gnome, the desktop file managers-GNU Midnight Commander and Nautilus- have a built-in FTP capability much like the KDE file manager. The FTP operation has been seamlessly integrated into standard desktop file operations. Downloading files from an FTP site is as simple as dragging files from one directory window to another, where one of the directories happens to be located on a remote FTP site. Use the Gnome file manager to access a remote FTP site, listing files in the remote directory, just as local files are. Just enter the FTP URL following the prefix **ftp://** and press ENTER. The top directory of the remote FTP site will be displayed. Simply use the file manager to progress through the remote FTP site's directory tree until you find the file you want. Then open another window for the local directory to which you want the remote files copied. In the window showing the FTP files, select those you want to download. Then use a CTRL-click and drag those files to the window for the local directory. A CTRL-click performs a copy operation, not a move. As files are downloaded, a dialog window appears showing the progress (see Figure 19-3).



Location: ftp://ftp.redhat.com/pub/redhat/tux/tux-2.0

tux-2.0
folder, 13 items
Fri, Feb 2 2001 at 12:00:00 AM

Location: /root/myprogs

myprogs
folder, 0 items
today at 2:29:12 AM

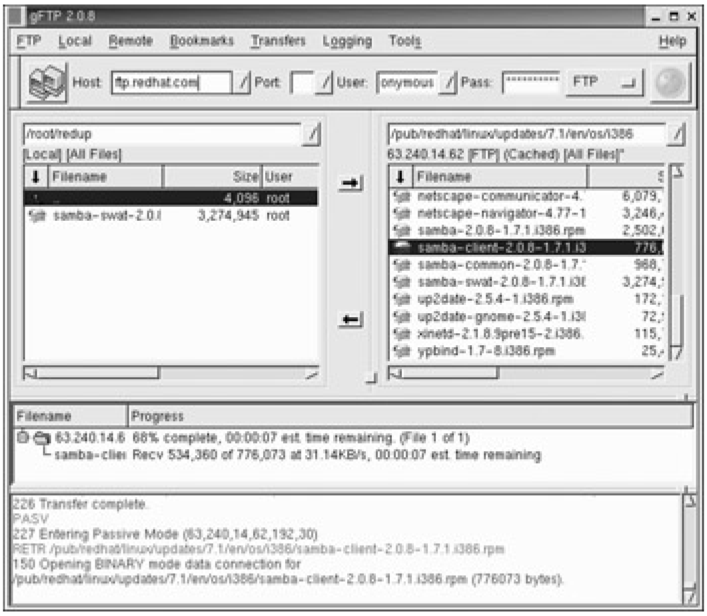
Files copied: 9 of 13

Copying: kernel-smp-2.4.3-0.99.20.tux.i686.rpm
From: /pub/redhat/tux/tux-2.0
To: /root/myprogs

Figure 19-3: Gnome Nautilus file manager performing FTP transfers
Note Nautilus will replace Midnight Commander in Gnome 1.4 and later versions of Red Hat.

gFTP

The gFTP program is a simpler Gnome FTP client designed to let you make standard FTP file transfers. It has an interface similar to WS_FTP used on Windows (see Figure 19-4). The gFTP window consists of several panes. The top-left pane lists files in your local directory, and the top-right pane lists your remote directory. Subdirectories have folder icons preceding their names. The parent directory can be referenced by the double period entry (**..**)with an up arrow at the top of each list. Double-click a directory entry to access it. The pathnames for all directories are displayed in boxes above each pane. You can enter a new pathname for a different directory to change to it, if you want.



Host: ftp.redhat.com Port: User: onymous Pass: ********** FTP

/root/redup
Local] [All Files]

```
..                    4,096  root
samba-swat-2.0.I  3,274,945  root
```

/pub/redhat/linux/updates/7.1/en/os/i386
63.240.14.62 [FTP] (Cached) [All Files]*

```
netscape-communicator-4.   6,079,
netscape-navigator-4.77-1  3,246,
samba-2.0.8-1.7.1.i386.rpm 2,502,
samba-client-2.0.8-1.7.1.i3  776,
samba-common-2.0.8-1.7.      968,
samba-swat-2.0.8-1.7.1.i3  3,274,
up2date-2.5.4-1.i386.rpm     172,
up2date-gnome-2.5.4-1.i3      72,
xinetd-2.1.8.9pre15-2.i386.  115,
ypbind-1.7-8.i386.rpm         25,
```

```
63.240.14.6  66% complete, 00:00:07 est. time remaining. (File 1 of 1)
└samba-clie  Recv 534,360 of 776,073 at 31.14KB/s, 00:00:07 est. time remaining
```

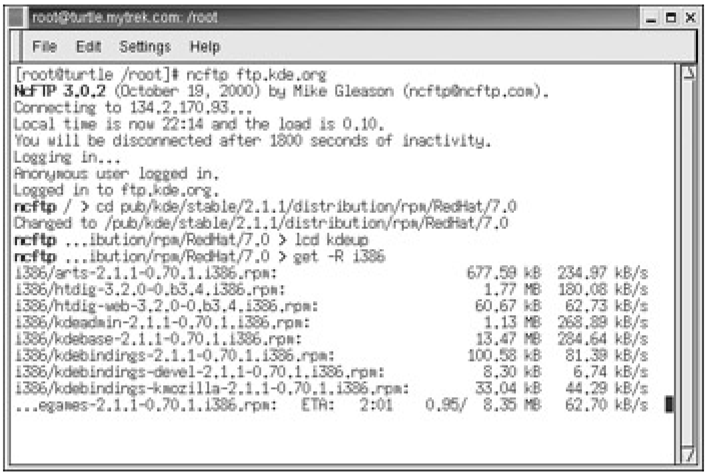
```
226 Transfer complete.
PASV
227 Entering Passive Mode (63,240,14,62,192,30)
RETR /pub/redhat/linux/updates/7.1/en/os/i386/samba-client-2.0.8-1.7.1.i386.rpm
150 Opening BINARY mode data connection for
/pub/redhat/linux/updates/7.1/en/os/i386/samba-client-2.0.8-1.7.1.i386.rpm (776073 bytes).
```

Figure 19-4: The gFTP Gnome FTP client

Two buttons between the panes are used for transferring files. The Left Arrow button, <-, downloads selected files in the remote directory, and the Right Arrow button, ->, uploads files from the local directory. To download a file, click it in the right-side pane and then click the Left Arrow button, <-. When the file is downloaded, its name appears in the left-side pane, your local directory. Menus across the top of the window can be used to manage your transfers. A connection manager enables you to enter login information about a specific site. You can specify whether to perform an anonymous login or to provide a user name and password. Click the Connect button to connect to that site. A drop-down menu for sites enables you to choose the site you want.

NcFTP

The NcFTP program, shown in Figure 19-5, has a screen-based interface that can be run from any shell command line. It does not use a desktop interface. FTP operations are executed using commands you enter at a prompt. Options and bookmarks can be selected using cursor-based menus. To start up NcFTP, you enter the **ncftp** command on the command line. If you are working in a window manager, such as KDE, Gnome, or FVWM, open a shell terminal window and enter the command at its prompt. The main NcFTP screen consists of an input line at the bottom of the screen with a status line above it. The remainder of the screen is used to display commands and responses from remote systems. For example, when you download files, a message specifying the files to be downloaded is displayed in the status line. NcFTP lets you set preferences for different features, such as anonymous login, progress meters, or a download directory. Enter the **pref** command to open the preferences screen. From there, you can select and modify the listed preferences.



```
[root@turtle /root]# ncftp ftp.kde.org
NcFTP 3.0.2 (October 19, 2000) by Mike Gleason (ncftp@ncftp.com).
Connecting to 134.2.170.93...
Local time is now 22:14 and the load is 0.10.
You will be disconnected after 1800 seconds of inactivity.
Logging in...
Anonymous user logged in.
Logged in to ftp.kde.org.
ncftp / > cd pub/kde/stable/2.1.1/distribution/rpm/RedHat/7.0
Changed to /pub/kde/stable/2.1.1/distribution/rpm/RedHat/7.0
ncftp ...ibution/rpm/RedHat/7.0 > lcd kdeup
ncftp ...ibution/rpm/RedHat/7.0 > get -R i386
i386/arts-2.1.1-0.70.1.i386.rpm:                      677.59 kB  234.97 kB/s
i386/htdig-3.2.0-0.b3.4.i386.rpm:                       1.77 MB  180.08 kB/s
i386/htdig-web-3.2.0-0.b3.4.i386.rpm:                  60.67 kB   62.73 kB/s
i386/kdeadmin-2.1.1-0.70.1.i386.rpm:                    1.13 MB  268.89 kB/s
i386/kdebase-2.1.1-0.70.1.i386.rpm:                    13.47 MB  284.64 kB/s
i386/kdebindings-2.1.1-0.70.1.i386.rpm:               100.58 kB   81.39 kB/s
i386/kdebindings-devel-2.1.1-0.70.1.i386.rpm:           8.30 kB    6.74 kB/s
i386/kdebindings-kmozilla-2.1.1-0.70.1.i386.rpm:       33.04 kB   44.29 kB/s
...egames-2.1.1-0.70.1.i386.rpm:   ETA:   2:01    0.95/  8.35 MB   62.70 kB/s
```

Figure 19-5: NcFTP

To connect to an FTP site, you enter the **open** command on the input line, followed by the site's address. The address can be either an IP address or a domain name, such as **ftp.gnome.org**. If you don't supply an address, then a list of your bookmarked sites is displayed, and you can choose one from there. By default, NcFTP attempts an anonymous login, using the term "anonymous" as your user name and your e-mail address as the password. When you successfully connect, the status bar displays the remote site's name on the left and the remote directory name on the right.

```
open ftp.gnome.org
```

If you want to log into a specific account on a remote site, have yourself prompted for the user name and password by using the **-u** option with the **open** command. The **open** command remembers the last kind of login you performed for a specific site and repeats it. If you want to change back to an anonymous login from a user login, you use the **-a** option with the **open** command. For busy sites, you may be unable to connect on the first try and you must repeat the open process. NcFTP has a redial capability you turn on with the **-r** option. The **-d** option sets the delay for the next attempt, and the **-g** option sets the maximum number of connection attempts. With the **lookup** command, you can obtain the IP and domain name addresses for an FTP site. The **lookup** command takes as an argument either the IP or domain name address and then displays both. This is useful for finding a site's IP address. With the **-v** option, more information, such as aliases, is retrieved. The NcFTP open options are shown in Table 19-2.

Table 19-2: NcFTP Open Options

Option	**Description**
-a	Connect anonymously
-u	Connect with user name and password prompts
-p *num*	Use specified port number when connecting
-r	Redial until connected
-d *num*	Set delay (*num*) in number of seconds for redial option
-g *num*	Specify the maximum number of redials

Once connected, you enter commands on the input line to perform FTP operations such as displaying file lists, changing directories, or downloading files. With the **ls** command, you can list the contents of the current remote directory. Use the **cd** command to change to another remote directory. The **dir** command displays a detailed listing of files. With the **page** command, you view the contents of a remote file, a screen at a time. To download files, you use the **get** command, and to upload files, you use the **put** command. During a download, a progress meter above the status bar displays how much of the file has been downloaded so far. The **get** command has several features described in more detail in the following section. When you finish, you can disconnect from the site with the **close** command. You can then use **open** to connect to another site, or quit the NcFTP program with the **quit** command. The **help** command lists all NcFTP commands. You can use the **help** command followed by the name of a command to display specific information on it.

The NcFTP program supports several commands that operate on your local system. These are usually standard FTP command names preceded by an *l*. **lcd** changes your local working directory, **lls** lists the contents of your local directory, **lpage** displays the contents of a local file a screen at a time, and **lpwd** displays the local directory's full pathname. For any other local commands or scripts you need to execute, use the shell escape command, **!**. Simply precede the shell command or script with an **!**.

The NcFTP program also provides commands for managing files and directories on your remote site, provided you have the permission to do so. You can use **mkdir** to create a remote directory, and **rmdir** to remove one. Use the **rm** command to erase remote files. With the **rename** command, you can change their names. The NcFTP commands are listed in Table 19-3.

Table 19-3: NcFTP Commands

Command	Description
help [*command*]	Lists names of NcFTP commands
cd [*directory*]	Changes the working directory on the remote host
create [*filename*]	Creates an empty file on the remote host, enabling you to use the filename as a message
debug	Turns debugging on or off
version	Displays version information
dir	Displays a detailed directory listing
echo	Displays a string, useful for macros
get	Downloads files from a remote host to your working directory
lcd [*directory*]	Changes the local working directory
lls	Lists files in your local working directory
lookup *host*	Looks up entries for remote hosts
lpage *filename*	Displays contents of local file, a page at time
lpwd	Displays the local current working directory
mkdir *directory name*	Creates a directory on the remote host
mode *mode*	Specifies transfer mode (*b* for block mode, *s* for stream mode)
open [*option*] *hostname*	Connects to a remote host. If no hostname is specified, the bookmark editor displays a host list from which you can choose one (**-a** forces anonymous login, **-u** forces user login, **-r** redials automatically, **-d** specifies time delay before redial-used with **-r;** and **-g** specifies the maximum number of redials-used with **-r**)
page *filename*	Displays the contents of a remote file
pdir	Same as **dir**, but outputs to your pager, enabling you to display a remote file list a page at a time. Used for command line interface
pls	Same as **ls**, but outputs to your pager. Used for command line interface
redir	Redisplays the last directory listing
predir	Redisplays the last directory listing and outputs to pager if working in command line interface
put *filename*	Uploads a file to a remote host
pwd	Displays the remote current working directory
rename *orig-name new-name*	Changes the name of a remote file
quit	Quits NcFTP
quote	Sends an FTP protocol command to the remote server
rhelp [*command*]	Sends a help request to the remote host

Table 19-3: NcFTP Commands

Command	Description
rm *filenames*	Erases remote files
rmdir *directories*	Removes remote directories
site *command*	Executes site-specific commands
type *type*	Changes transfer type (ASCII, binary, image)
! *command*	Escapes to the shell and executes the following shell command or script

The NcFTP program also has a colon mode of operation that enables you to issue a single **ncftp** command to download a file. Enter the **ncftp** command, followed by a URL, for the file you want. You can enter the command on the shell command line or place it within a script. For example, the following command downloads the **readme** file on the Red Hat FTP site:

```
$ ncftp ftp.redhat.com/pub/README
```

In the colon mode, the **-c** option sends the file to the standard output and the **-m** option pipes it to your pager, usually the **more** program.

```
$ ncftp -c ftp.redhat.com/pub/README > ~/redhatinfo/readme
$ ncftp -m ftp.redhat.com/pub/README
```

NcFTP Download Features

The NcFTP **get** command differs significantly from the original FTP client's **get** command. Whereas the original FTP client uses two commands, **get** and **mget**, to perform download operations, NcFTP uses only the **get** command. However, the NcFTP **get** command combines the capabilities of both **mget** and **get** into the **get** command, as well as adding several new features. Table 19-4 lists the various **get** command options. By default, the NcFTP **get** command performs wildcard matching for filenames. If you enter only part of a filename, the **get** command tries to download all files beginning with that name. You can turn off wildcard matching with the **-G** option, in which case you must enter the full names of the files you want. The following example downloads all files with names beginning with "Xfree86" and is similar to using **mget Xfree86*** in the original FTP:

Table 19-4: NcFTP get Options

Command	Description
-G	Turn wildcard matching for filenames on or off
-R *directory*	Download a directory and all its subdirectories (recursive)
-f *filenames*	Force the download of all specified files, even if older or the same as local ones
-C	Force resumption of a download from where it was interrupted
-z *remote-file local-file*	Rename a remote file on your local system
-n *num*	Download files no older than the specified number of days

```
get Xfree86
```

The **get** command checks to see if you already have a file you are trying to download. If so, it skips the download. The **get** command also checks if the file you already have is a newer version, in which case it also skips the download. This is a helpful feature for easily maintaining upgrade files. You can simply access the update directory on the remote site, and then use the **get** command with the * to download to the directory you are using to keep your upgrade file. Only newer versions or newly added upgrade files are downloaded, instead of the entire set. If you want to download a file, even though you have it already, you can force the download with the **-f** option. For example, to download upgrades for Red Hat manually, you can connect to the Red Hat upgrade directory in the Red Hat FTP site and then issue the following **get** command:

```
get *
```

Note If you were interrupted during a download, you can restart the download from where you left off. This feature is built into NcFTP. (On other FTP programs, it can be invoked with the **reget** command.) NcFTP checks to see if you have already started to download a file and then continues from where you left off.

Certain features require you to enter an option on the command line after the **get** command. For example, adding the **-R** command specifies a recursive capability, enabling you to download and create subdirectories and their files. This command is particularly helpful in downloading upgrade directories, such as Red Hat's, which contain several subdirectories. The following example downloads the **i386** directory and all its subdirectories:

```
get -R i386
```

If you want to give a file a different name on your local system, use the **-z** option. Enter the local filename you want after the remote filename. The following example downloads the **readme** file and renames it **calinfo**. If you did not use the **-z** option, then both names would be taken as files to be downloaded, instead of only the first.

```
get -z readme calinfo
```

To obtain recent files only, you can use the **-n** option, which takes as its argument a number of days. Files older than the specified number of days are not retrieved. The following example downloads files posted within the last 30 days:

```
get -n 30 *
```

Bookmarks and Macros

When you disconnect (close) from a site, NcFTP automatically saves information about it. This includes the site address, the directory you were in, and the login information. This information is placed in a file called **bookmarks** in your **.ncftp** directory. The site information is given a bookmark name you can use to access the site easily again. The bookmark name is usually the key name in the site's address. You can use this name to connect to the site. For example, ftp.redhat.com could be named **redhat**. You could then connect to it with the command

```
open redhat
```

You can edit your bookmark entries using the bookmark editor. Enter the command **bookmarks** to bring up the editor. Remote systems you have accessed are listed on the right side of the screen. Bookmark commands are listed on the left. You can change the bookmark name or edit login information, such as the user name or password, the remote directory, or the transfer mode.

The NcFTP program supports macros for simple operations. You create macros by entering macro definitions in the macros file located in your **.ncftp** directory. Initially, no such file will exist, so you have to create one using any text editor. The **macros** file is a simple text file you can edit with any text editor. The syntax for a macro definition follows:

```
macro macro-name
 ftp-commands
end
```

A macro executes NcFTP commands. Remember, however, the **!** is an NcFTP command that enables you to execute any Linux command or script. With a preceding **!** you can define an NcFTP macro that executes any shell command or any script you have written. A simple example of a macro is

```
macro ascii
 type ascii
end
```

Macros support parameters similar to those used by shell programs. Arguments entered after a macro name can be referenced in the macro using a **$** sign and the number of the argument in the argument list. **$1** references the first argument, **$2** the second, and so on. **$*** is a special parameter that references all arguments, and **$@** references all arguments, encasing each in double quotes.

```
macro cdls
 cd $1
 ls
end
```

The NcFTP program also supports a limited number of event macros. These are macros executed when a certain event is detected, such as when the program starts or shuts down. For example, a macro defined with the name **.start.ncftp** has its commands executed every time you start NcFTP; **.quit.ncftp** executes its commands when you quit. Site-specific macros also execute whenever it is necessary to access or disconnect from certain sites. These macros begin with either the open or close event, followed by the site's bookmark. For example, a macro defined with the name **.open.redhat** would execute its commands whenever you connected to the Red Hat site. A macro named **.open.any** has its commands executed whenever you connect to any site, and one named **.close.any** executes whenever you disconnect from a site.

ftp

The name **ftp** designates the original FTP client used on Unix and Linux systems. **ftp** uses a command line interface, and it has an extensive set of commands and options you can use to manage your FTP transfers. You start the **ftp** client by entering the command **ftp** at a shell prompt. If you have a specific site you want to connect to, you can include the name of that site on the command line after the **ftp** keyword. Otherwise, you need to connect to the remote system with the **ftp** command **open**. You are then prompted for the name of the remote system with the prompt "(to)". Upon entering the remote system name, **ftp** connects you to the system and then prompts you for a login name. The prompt for the login name consists of the word "Name" and, in parentheses, the system name and your local login name. Sometimes the login name on the remote system is the same as the login name on your own system. If the names are the same, press ENTER at the prompt. If they are different, enter the remote system's login name. After entering the login name, you are prompted for the password. In the next example, the user connects to the remote system **garnet** and logs into the **robert** account:

```
$ ftp
ftp> open
(to) garnet
Connected to garnet.berkeley.edu.
220 garnet.berkeley.edu FTP server ready.
Name (garnet.berkeley.edu:root): robert
password required
Password:
user robert logged in
ftp>
```

Once logged in, you can execute Linux commands on either the remote system or your local system. You execute a command on your local system in **ftp** by preceding the command with an exclamation point. Any Linux commands without an exclamation point are executed on the remote system. One exception exists to this rule. Whereas you can change directories on the remote system with the **cd** command, to change directories on your local system, you need to use a special **ftp** command called **lcd** (local **cd**). In the next example, the first command lists files in the remote system, while the second command lists files in the local system:

```
ftp> ls
ftp> !ls
```

The **ftp** program provides a basic set of commands for managing files and directories on your remote site, provided you have the permission to do so (see Table 19-5). You can use **mkdir** to create a remote directory, and **rmdir** to remove one. Use the **delete** command to erase a remote file. With the **rename** command, you can change the names of files. You close your connection to a system with the **close** command. You can then open another connection if you want. To end the **ftp** session, use the **quit** or **bye** command.

Table 19-5: ftp Client Commands

Command	**Effect**
ftp	Invokes **ftp** program
open *site-address*	Opens a connection to another system
close	Closes connection to a system
quit or **bye**	Ends **ftp** session
ls	Lists the contents of a directory
dir	Lists the contents of a directory in long form
get *filename*	Sends file from remote system to local system

Table 19-5: ftp Client Commands

Command	**Effect**
put *filename*	Sends file from local system to remote system
mget *regular-expression*	Enables you to download several files at once from a remote system; you can use special characters to specify the files; you are prompted one by one, in turn, for each file transfer
mput *regular-expression*	Enables you to send several files at once to a remote system; you can use special characters to specify the files; you are prompted one by one for each file to be transferred
runique	Toggles storing of files with unique filenames. If a file already exists with the same filename on the local system, a new filename is generated
reget *filename*	Resumes transfer of an interrupted file from where you left off
binary	Transfers files in binary mode
ascii	Transfers files in ASCII mode
cd *directory*	Changes directories on the remote system
lcd *directory*	Changes directories on the local system
help or **?**	Lists **ftp** commands
mkdir *directory*	Creates a directory on the remote system
rmdir	Deletes a remote directory
delete *file name*	Deletes a file on the remote system
mdelete *file-list*	Deletes several remote files at once
rename	Renames a file on a remote system
hash	Displays progressive hash signs during download
status	Displays current status of **ftp**

```
ftp> close
ftp> bye
Good-bye
$
```

File Transfer

To transfer files to and from the remote system, use the **get** and **put** commands. The **get** command receives files from the remote system to your local system, and the **put** command sends files from your local system to the remote system. In a sense, your local system gets files *from* the remote and puts files *to* the remote. In the next example, the file **weather** is sent from the local system to the remote system using the **put** command:

```
ftp> put weather
PORT command successful.
ASCII data connection
ASCII Transfer complete.
ftp>
```

If a download is ever interrupted, you can resume the download with **reget**. This is helpful for an extremely large file. The download resumes from where it left off, so the whole file needn't be downloaded again. Also, be sure to download binary files in binary mode. For most FTP sites, the binary mode is the default, but some sites might have ASCII (text) as the default. The command **ascii** sets the character mode, and the command **binary** sets the binary mode. Most software packages available at Internet sites are archived and compressed files, which are binary files. In the next example, the transfer mode is set to binary, and the archived software package **mydata.tar.gz** is sent from the remote system to your local system using the **get** command:

```
ftp> binary
ftp> get mydata.tar.gz
PORT command successful.
Binary data connection
Binary Transfer complete.
ftp>
```

You may often want to send several files, specifying their names with wildcard characters. The **put** and **get** commands, however, operate only on a single file and do not work with special characters. To transfer several files at a time, you have to use two other commands, **mput** and **mget**. When you use **mput** or **mget**, you are prompted for a file list. You can then either enter the list of files or a file-list specification using special characters. For example, ***.c** specifies all the files with a **.c** extension, and * specifies all files in the current directory. In the case of **mget**, each file is sent, one by one, from the remote system to your local system. Each time, you are prompted with the name of the file being sent. You can type **y** to send the file or **n** to cancel the transmission. You are then prompted for the next file. The **mput** command works in the same way, but it sends files from your local system to the remote system. In the next example, all files with a **.c** extension are sent to your local system using **mget**:

```
ftp> mget
(remote-files) *.c
mget calc.c? y
PORT command successful
ASCII data connection
ASCII transfer complete
mget main.c? y
PORT command successful
ASCII data connection
ASCII transfer complete
ftp>
```

Answering the prompt for each file can be a tedious prospect if you plan to download a large number of files, such as those for a system update. In this case, you can turn off the prompt with the **prompt** command, which toggles the interactive mode on and off. The mget operation then downloads all files it matches, one after the other.

```
ftp> prompt
Interactive mode off.
ftp> mget
(remote-files) *.c
 PORT command successful
ASCII data connection
ASCII transfer complete
PORT command successful
```

```
ASCII data connection
ASCII transfer complete
ftp>
```

Note To access a public FTP site, you have to perform an anonymous login. Instead of a login name, you enter the keyword **anonymous** (or **ftp**). Then, for the password, you enter your Internet address. Once the **ftp** prompt is displayed, you are ready to transfer files. You may need to change to the appropriate directory first or set the transfer mode to binary.

Automatic Login and Macros: .netrc

The **ftp** client has an automatic login capability and support for macros. Both are entered in a user's **ftp** configuration file called **.netrc**. Each time you connect to a site, the **.netrc** file is checked for connection information, such as a login name and password. In this way, you needn't enter a login name and password each time you connect to a site. This feature is particularly useful for anonymous logins. Instead of your having to enter the user name "anonymous" and your e-mail address as your password, they can be automatically read from the **.netrc** file. You can even make anonymous login information your default so that, unless otherwise specified, an anonymous login is attempted for any FTP site to which you try to connect. If you have sites you must log into, you can specify them in the **.netrc** file and, when you connect, either automatically log in with your user name and password for that site or be prompted for them.

Entries in the **.netrc** file have the following syntax. An entry for a site begins with the term "machine," followed by the network or Internet address, and then the login and password information.

```
 machine system-address login remote-login-name password password
```

The following example shows an entry for logging into the **dylan** account on the **turtle.trek.com** system:

```
 machine golf.mygames.com login dylan password legogolf
```

For a site you would anonymously log into, you enter the word "anonymous" for the login name and your e-mail address for the password.

```
 machine ftp.redhat.com login anonymous password dylan@turtle.trek.com
```

In most cases, you are using **ftp** to access anonymous FTP sites. Instead of trying to make an entry for each one, you can make a default entry for anonymous FTP login. When you connect to a site, **ftp** looks for a machine entry for it in the **.netrc** file. If none exists, then **ftp** looks for a default entry and uses that. A default entry begins with the term "default" with no network address. To make anonymous logins your default, enter anonymous and your e-mail address as your login and password.

```
 default login anonymous password
dylan@turtle.trek.com
```

A sample **.netrc** file with a machine definiton and a default entry is shown here.

.netrc

```
machine golf.mygames.com login dylan password legogolf
default login anonymous password dylan@turtle.trek.com
```

You can also define macros in your **.netrc** file. With a macro, you can execute several ftp operations at once using only the macro name. Macros remain in effect during a connection. When you close a connection, the macros are undefined. Although a macro can be defined on your **ftp** command line, defining them in **.netrc** entries makes more sense. This way, you needn't redefine them again. They are read automatically from the **.netrc** file and defined for you. You can place macro definitions within a particular machine entry in the **.netrc** file or in the default entry. Macros defined in machine entries are defined only when you connect to that site. Macros in the default entry are defined whenever you make a connection to any site.

The syntax for a macro definition follows. It begins with the keyword **macdef**, followed by the macro name you want to give it, and ends with an empty line. **ftp** macros can take arguments, referenced within the macro with **$n**, where **$1** references the first argument, and **$2** the second, and so on. If you need to use a **$** character in a macro, you have to quote it using the backslash, **\$**.

```
macdef macro-name
ftp commands
empty-line
```

The **redupd** macro, defined next, changes to a directory where it then downloads Red Hat updates for the current release. It also changes to a local directory where the update files are to be placed. The **prompt** command turns off the download prompts for each file. The **mget** command then downloads the files. The macro assumes you are connected to the Red Hat FTP site.

```
defmac redupd
cd pub/redhat/current
lcd /root/redupdate
prompt
mget *
```

A sample **.netrc** file follows with macros defined for both specific and default entries. An empty line is placed after each macro definition. You can define several macros for a machine or the default entry. The macro definitions following a machine entry up to the next machine entry are automatically defined for that machine connection.

.netrc

```
machine updates.redhat.com login anonymous password
dylan@turtle.trek.com
# define a macro for downloading updated from the Red Hat site
defmac redupd
cd pub/redhat/current
lcd /root/redupdate
prompt
mget *

default login anonymous password dylan@turtle.trek.com
```

```
defmac lls
!ls
```

Chapter 20: The World Wide Web and Java

Overview

The World Wide Web (WWW, or the Web) is a hypertext database of different types of information, distributed across many different sites on the Internet. A *hypertext database* consists of items linked to other items, which, in turn, may be linked to yet other items, and so on. Upon retrieving an item, you can use that item to retrieve any related items. For example, you could retrieve an article on the Amazon rain forest and then use it to retrieve a map or a picture of the rain forest. In this respect, a hypertext database is like a web of interconnected data you can trace from one data item to another. Information is displayed in pages known as *Web pages.* On a Web page, certain keywords are highlighted that form links to other Web pages or to items, such as pictures, articles, or files.

The Web links data across different sites on the Internet throughout the world. The Web originated in Europe at CERN research laboratories, and CERN remains the original Web server. An Internet site that operates as a Web server is known as a *Web site.* Such Web sites are often dedicated to specialized topics or institutions-for example, the Smithsonian Web site or the NASA Web site. These Web sites usually have an Internet address that begins with "www", as in **www.redhat.com**, the Web site for Red Hat, Inc. Once connected to a Web site, you can use hypertext links to move from one Web page to another.

To access the Web, you use a client program called a *browser.* You can choose from many different Web browsers. Browsers are available for use on Unix, Windows, the Mac, and Linux. Certain browsers, such as Netscape and Mozilla, have versions that operate on all such systems. On your Linux system, you can choose from several Web browsers, including Netscape Navigator. Navigator is available as part of all Linux distributions. Netscape and Mozilla are X Windows-based browsers that provide full picture, sound, and video display capabilities. Most distributions also include the Lynx browser, a line-mode browser that displays only lines of text. The K Desktop incorporates Web browser capabilities into its file manager, letting a directory window operate as a Web browser. Gnome-based browsers, such as Express and Mnemonic, are also designed to be easily enhanced.

URL Addresses

An Internet resource is accessed using a Universal Resource Locator (URL). A URL is composed of three elements: the transfer protocol, the hostname, and the pathname. The transfer protocol and the hostname are separated by a colon and two slashes, **://**. The *pathname* always begins with a single slash.

```
transfer-protocol://host-name/path-name
```

The *transfer protocol* is usually HTTP (Hypertext Transfer Protocol), indicating a Web page. Other possible values for transfer protocols are gopher, ftp, and file. As their names suggest, gopher and ftp initiate Gopher and FTP sessions, whereas file displays a local file on your own system, such as a text or an HTML file. Table 20-1 lists the various transfer protocols.

Table 20-1: Web Protocols

Protocol	**Description**
http	Hypertext Transfer Protocol for Web site access
gopher	Access Gopher site
ftp	File Transfer Protocol for anonymous FTP connections
telnet	Makes a telnet connection
wais	Access WAIS site
news	Reading Usenet news; uses Net News Transfer Protocol (NNTP)

The *hostname* is the computer on which a particular Web site is located. You can think of this as the address of the Web site. By convention, most hostnames begin with www. In the next example, the URL locates a Web page called guides.html on the **www.kernel.org** Web site in the LDP directory:

```
http://www.kernel.org/LDP/guides.html
```

If you do not want to access a particular Web page, you can leave the file reference out, and then you automatically access the Web site's home page. To access a Web site directly, use its hostname. If no home page is specified for a Web site, the file **index.html** in the top directory is often used as the home page. In the next example, the user brings up the Red Hat home page:

```
http://www.redhat.com/
```

The pathname specifies the directory where the resource can be found on the host system, as well as the name of the resource's file. For example, **/pub/Linux/newdat.html** references an HTML document called newdat located in the **/pub/Linux** directory. As you move to other Web pages on a site, you may move more deeply into the directory tree. In the following example, the user accesses the FAQ.html document in the directory **support/docs/faqs/rhl_general_faq/FAQ.html/**:

```
http://
http://www.redhat.com/support/docs/faqs/rhl_general_faq/FAQ.html
```

As just explained, if you specify a directory pathname without a particular Web page file, the Web site looks for a file called **index.html** in that directory. An **index.html** file in a directory operates as the default Web page for that directory. In the next example, the **index.html** Web page in the /**apps/support** directory is displayed:

```
http://www.redhat.com/apps/support/index.html
```

You can use this technique to access local Web pages on your system. For example, once installed, the demo Web pages for Java are located in **/usr/local/java/**. Because this is on your local system, you needn't include a hostname. An **index.html** page in the **/usr/ local/java/** directory is automatically displayed when you specify the directory path. You can do the same for your system documentation, which, on most distributions, is in Web page format located in the **/usr/doc/HTML/ldp** directory.

```
file:/usr/local/java
file:/usr/doc/HTML/ldp
```

Table 20-2: Web File Types

File Type	**Description**
.html	Web page document formatted using HTML, the Hypertext Markup Language
Graphics Files	
.gif	Graphics, using GIF compression
.jpeg	Graphics, using JPEG compression
Sound Files	
.au	Sun (UNIX) sound file
.wav	Microsoft Windows sound file
.aiff	Macintosh sound file
Video Files	
.QT	QuickTime video file, multiplatform
.mpeg	Video file
.avi	Microsoft Windows video file

If you reference a directory that has no **index.html** file, the Web server creates one for you, and your browser then displays it (see Chapter 25). This index simply lists the different files and directories in that directory. You can click an entry to display a file or to move to another directory. The first entry is a special entry for the parent directory.

The resource file's extension indicates the type of action to be taken on it. A picture has a **.gif** or **.jpeg** extension and is converted for display. A sound file has a **.au** or **.wav** extension and is played. The following URL references a **.gif** file. Instead of displaying a Web page, your browser invokes a graphics viewer to display the picture. Table 20-2 provides a list of the different file extensions.

```
http://www.train.com/engine/engine1.gif
```

Web Pages

A Web page is a specially formatted document that can be displayed by any Web browser. You can think of a Web page as a word processing document that can display both text and graphics. Within the Web page, links can be embedded that call up other Internet resources. An Internet resource can be a graphic, a file, a telnet connection, or even another Web page. The Web page acts as an interface for accessing different Internet tools, such as FTP to download files or telnet to connect to an online catalog or other remote service.

Web pages display both text and graphics. Text is formatted with paragraphs and can be organized with different headings. Graphics of various sizes may be placed anywhere in the page. Throughout the page there are usually links you can use to call up other Internet resources. Each *link* is associated with a particular Internet resource. One link may reference a picture; another, a file. Others may reference other Web pages or even other Web sites. These links are specially highlighted text or graphics that usually appear in a different color from the rest of the text. Whereas ordinary text may be black, text used for links might be green, blue, or red. You select a particular link by moving your mouse pointer to that text or picture, and then clicking it. The Internet resource associated with that link is then called up. If the resource is a picture, the picture is displayed. If it is another Web page, that Web page is displayed. If the Internet resource is on another Web site, that site is accessed. The color of a link indicates its status and the particular Web browser you are using. Both Mozilla and Netscape browsers by default use blue for links you have not yet accessed. Both use the color purple for links you have already accessed. All these colors can be overridden by a particular Web page.

Your Web browser keeps a list of the different Web pages you access for each session. You can move back and forth easily in that list. Having called up another Web page, you can use your browser to move back to the previous one. Web browsers construct their lists according to the sequence in which you displayed your Web pages. They keep track of the Web pages you are accessing, whatever they may be. On many Web sites, however, several Web pages are meant to be connected in a particular order, like chapters in a book. Such pages usually have buttons displayed at the bottom of the page that reference the next and previous pages in the sequence. Clicking Next displays the next Web page for this site. The Home button returns you to the first page for this sequence.

Web Browsers

Most Web browsers are designed to access several different kinds of information. Web browsers can access a Web page on a remote Web site or a file on your own system. Some browsers can also access a remote news server or an FTP site. The type of information for a site is specified by the keyword **http** for Web sites, **nntp** for news servers, **ftp** for FTP sites, and **file** for files on your own system.

To access a Web site, you enter **http://** followed by the Internet address of the Web site. If you know a particular Web page you want to access on that Web site, you can add the pathname for that page, attaching it to the Internet address. Then simply press ENTER. The browser connects you to that Web site and display its home page or the page you specified.

You can just as easily use a Web browser to display Web pages on your own system by entering the term **file** followed by a colon, **file:**, with the pathname of the Web page you want to display. You do not specify an Internet site. Remember, all Web pages usually have the extension **.html**. Links within a Web page on your own system can connect you to other Web pages on your system or to Web pages on remote systems. When you first start a Web browser, your browser displays a local Web page on your own system. The default page on Red Hat is a local page with links to the Red Hat Web site where you can obtain online support. If you want, you can create your own Web pages, with their own links, and make one of them your default Web page.

Web pages on a Web site often contain links to other Web pages, some on the same site and others at other Web sites. Through these links, you can move from one page to another. As you move from Web page to Web page using the links or buttons, your browser displays the URL for the current page. Your browser keeps a list of the different Web pages you have accessed in a given session. Most browsers have buttons that enable you to move back and forth through this list. You can move to the Web page you displayed before the current one and then move back further to the previous one. You can move forward again to the next page, and so on.

To get to a particular page, you may have moved through a series of pages, using links in each to finally reach the Web page you want. To access any Web page, all you need is its URL address. If you want to access a particular page again, you can enter its URL address and move directly to it, without moving through the intervening pages as you did the first time. Instead of writing down the URL addresses and entering them yourself, most Web browsers can keep *bookmarks*-a list of favorite Web pages you want to access directly. When you are displaying a Web page you want to access later, instruct your browser to place it on the Bookmarks list. The Web page is usually listed in the Bookmarks list by its title, not its URL. To access that Web page later, select the entry in the Bookmark list.

Most Web browsers can also access FTP and Gopher sites. You may find using a Web browser to access an FTP site with anonymous access is easier than using an FTP client. Directories and files are automatically listed, and selecting a file or directory is only a matter of clicking its name. First enter **ftp://** and then the Internet address of the FTP site. The contents of a directory are then displayed, listing files and subdirectories. To move to another directory, just click it. To download a file, click its name. You see an entry listed as double periods (**..**), representing the parent directory. You can move down the file structure from one subdirectory to another and move back up one directory at a time by selecting double periods (**..**). To leave the FTP site, return to your own home page. You can also use your browser to access Gopher sites. Enter **gopher://** followed by the Internet address of the Gopher site. Your Web browser then displays the main Gopher menu for that site, and you can move from one Gopher menu to the next.

Most browsers can connect to your news server to access specified newsgroups or articles. This is a local operation, accessing the news server to which you are already connected. You enter **nntp** followed by a colon and the newsgroup or news article. Some browsers, such as Netscape, have an added newsreader browser that allows them to access any remote news servers.

As noted previously, several popular browsers are available for Linux. Three distinctive ones are described here: Netscape Navigator, Konqueror, and Lynx. Netscape is an X Windows-based Web browser capable of displaying graphics, video, and sound, as well as operating as a newsreader and mailer. Konqueror is the K Desktop file manager. KDE has integrated full Web-browsing capability into the Konqueror file manager, letting you seamlessly access the Web and your file system with the same application. Lynx and Links are command line-based browsers with no graphics capabilities but in every other respect are fully functional Web browsers.

Netscape Navigator and Mozilla

Hypertext databases are designed to access any kind of data, whether it is text, graphics, sound, or even video. Whether you can actually access such data depends to a large extent on the type of browser you use. Mozilla is a browser based on the Netscape core source code known as mozilla. In 1998, Netscape made this source code freely available under the Netscape Public License (NPL). The Mozilla Project based at www.mozilla.org is developing a commercial-level browser based on mozilla source code. Mozilla is developed on an open source model much like Linux, KDE, and Gnome. Developers can submit modifications and additions over the Internet to the Mozilla Web site. Mozilla releases are referred to as Milestones. Mozilla is currently released under both the NPL license for modifications of mozilla code and the MPL license (Mozilla Public License) for new additions. In future releases, Red Hat will use Mozilla as its primary browser, in place of Netscape.

Mozilla is an X Windows application you operate from your desktop. Red Hat has a Mozilla entry in the desktop's Internet menu. Mozilla displays an area at the top of the screen for entering a URL address and a series of buttons for various Web page operations. Drop-down menus provide access to Mozilla features. To access a Web site, you enter its address in the URL area and press ENTER. The icon bar across the top of the browser holds buttons for moving from one page to another and performing other operations (see Figure 20-1).

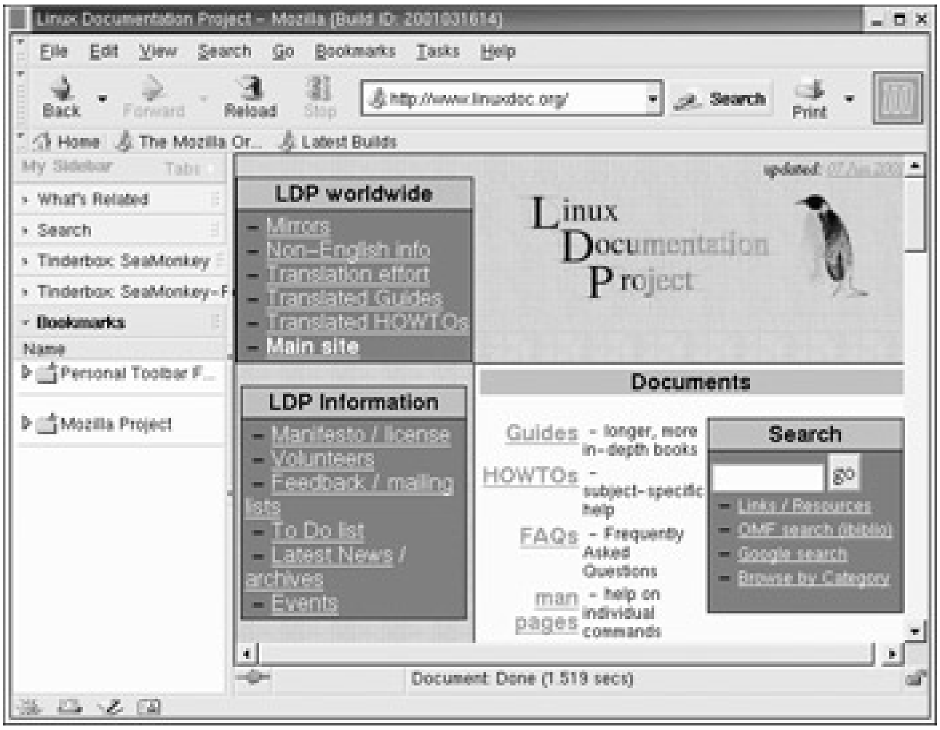


Figure 20-1: Mozilla Web browser

Mozilla refers to the URLs of Web pages you want to keep in a hotlist as bookmarks, marking pages you want to access directly. The Bookmarks menu enables you add your favorite Web pages to a hotlist. You can then view your bookmarks and select one to view. You can also edit your list of bookmarks, adding new ones or removing old ones. History is a list of previous URLs you have accessed. If you want to return to a Web page you did not save as a bookmark, you can find it in the History list. Additionally, you can use Mozilla to receive and send mail, as well as to access Usenet newsgroups.

The Options menu in Mozilla enables you to set several different kinds of preferences for your browser. You can set preferences for mail and news, the network, and security, as well as general preferences. In general preferences, you can determine your home page and how you want the toolbar displayed. In the Mail/News Account Settings, you can enter the mail and news servers you use on the Internet. Mozilla can be set to access any number of news servers you subscribe to that use the NNTP transfer protocols. You can switch from one news server to another if you want.

If you are on a network that connects to the Internet through a firewall, you must use the Proxies screen to enter the address of your network's firewall gateway computer. A *firewall* is a computer that operates as a controlled gateway to the Internet for your network. Several types of firewalls exist. One of the most restrictive uses programs called *proxies,* which receive Internet requests from users and then make those requests on their behalf. There is no direct connection to the Internet. From the Options menu, select Network and then choose the Proxies screen. Here, enter the IP address of your network's firewall gateway computer.

Through the Mail item in the Tasks menu, you can open a fully functional mail client with which you can send and receive messages over the Internet. The News item, also in the Tasks menu, opens a fully functional newsreader with which you can read and post articles in Usenet newsgroups. In this respect, your Mozilla is more than just a Web browser. It is also a mail program and a newsreader.

One of the more popular Web browsers is Netscape Navigator. Versions of Netscape operate on different graphical user interfaces such as X Windows, Microsoft Windows, and the Macintosh. Using X Windows, the Netscape browser can display graphics, sound, video, and Java-based programs (you learn about Java a little later in the chapter). You can obtain more information about Netscape on its Web site: **www.netscape.com**. Netscape Navigator is now included with Red Hat Linux 7.1, but will be replaced in future releases with Mozilla. You can obtain more recent versions from the Red Hat distribution FTP site at **ftp.redhat.com**.

K Desktop File Manager: Konqueror

If you are using the K Desktop, then you can use a file manager window as a Web browser, as shown in Figure 20-2. The K Desktop's file manager is automatically configured to act as a Web browser. It can display Web pages, including graphics and links. The K Desktop's file manager supports standard Web page operation, such as moving forward and backward through accessed pages. Clicking a link accesses and displays the Web page referenced. In this respect, the Web becomes seamlessly integrated into the K Desktop.

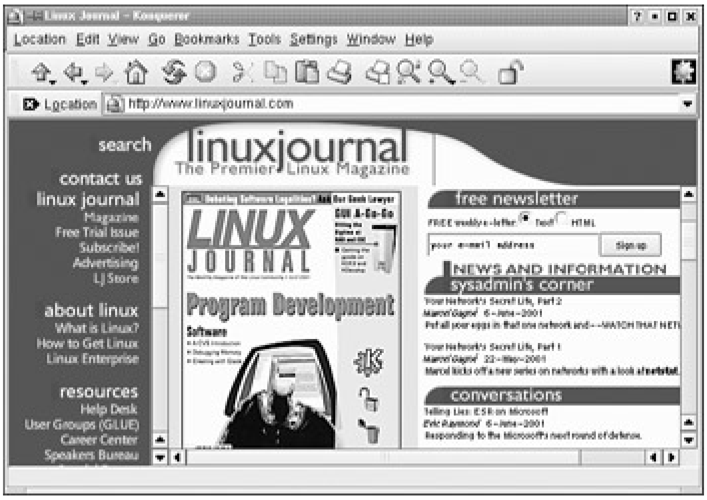


Figure 20-2: The K Desktop file manager as a Web browser

Gnome Web Browsers: Nautilus, Galeon, Express, and Mnemonic

The new Gnome file manager, Nautilus, used in Gnome 1.4, is a functional Web browser, just like Konqueror (see Figure 8-11 in Chapter 8). In the Nautilus location box you can enter a

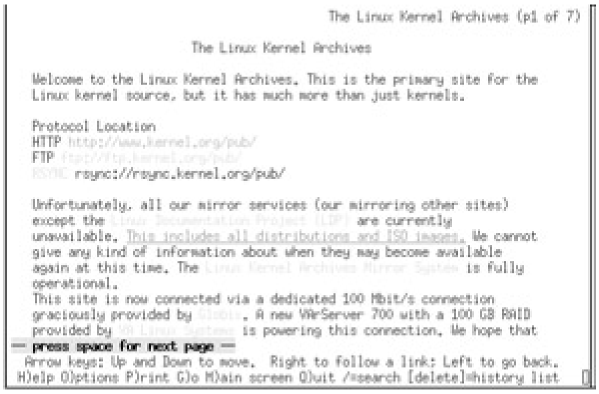
Web address and Nautilus will access and display that Web page. The file manager Forward and Backward buttons, as well as bookmarks, help you navigate through previously viewed pages. However, it is not a fully functional Web browser. Nautilus will display icons in its sidebar for dedicated Web browsers installed on your system. Click on one to start using that Web browser instead of Nautilus.

Note Midnight Commander, the Gnome 1.2 file manager used in Red Hat 7.1, does not have Web browser capability.

Several other Gnome-based Web browsers are also available. Galeon, Express, and Mnemonic support standard Web operations. Galeon is a Gnome Web browser designed to be fast with a very light interface. It supports drag-and-drop operations, multiple selections, and bookmark imports. It is based on Gecko (the mozilla rendering engine). You can find out more about Galeon at **galeon.sourceforge.net**. Express is designed to rely on plug-ins for Web features. This way, the browser can be made as complex or simple as you want. All major operations, such as viewers and protocols, are handled as plug-ins. This design allows new features to be easily added in this way. Mnemonic is an extensible and modular Web browser.

Lynx: Line-Mode Browser

Lynx is a line-mode browser you can use without X Windows. A Web page is displayed as text only (see Figure 20-3). A text page can contain links to other Internet resources, but does not display any graphics, video, or sound. Except for the display limitations, Lynx is a fully functional Web browser. You can use Lynx to download files or to make telnet connections. All information on the Web is still accessible to you. Because it does not require much of the overhead that graphics-based browsers need, Lynx can operate much faster, quickly displaying Web page text. To start the Lynx browser, you enter **lynx** on the command line and press ENTER.



```
                                      The Linux Kernel Archives (p1 of 7)

                    The Linux Kernel Archives

   Welcome to the Linux Kernel Archives. This is the primary site for the
   Linux kernel source, but it has much more than just kernels.

   Protocol Location
   HTTP http://www.kernel.org/pub/
   FTP ftp://ftp.kernel.org/pub/
   RSYNC rsync://rsync.kernel.org/pub/

   Unfortunately, all our mirror services (our mirroring other sites)
   except the Linux Documentation Project (LDP) are currently
   unavailable. This includes all distributions and ISO images. We cannot
   give any kind of information about when they may become available
   again at this time. The Linux Kernel Archives Mirror System is fully
   operational.
   This site is now connected via a dedicated 100 Mbit/s connection
   graciously provided by Globix. A new VArServer 700 with a 100 GB RAID
   provided by VA Linux Systems is powering this connection. We hope that
 -- press space for next page --
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

Figure 20-3: Lynx Web browser

The links are displayed in bold and dispersed throughout the text of the Web page. A selected link is highlighted in reverse video with a shaded rectangle around the link text. The first link is automatically selected. You can then move sequentially from one link to the next on a page by pressing the DOWN ARROW key. The UP ARROW key moves you back to a previous link. To choose a link, first highlight it and then press either ENTER or the RIGHT ARROW key. If you want to go to a specific site, press G. This opens a line at the bottom of the screen with the prompt **URL to open:**. There, you can enter the URL for the site you want. Pressing M returns you to your home page. The text of a Web page is displayed one screen at a time.

To move to the next screen of text, you can either press SPACEBAR or PAGE DOWN. PAGE UP displays the previous screen of text. Pressing DOWN ARROW and UP ARROW moves to the next or previous links in the text, displaying the full screen of text around the link. To display a description of the current Web page with its URL, press the = key.

Note Another useful text-based browser shipped with Red Hat Linux is **links**. **links** provides frame and table support.

Lynx uses a set of one-letter commands to perform various browser functions. By pressing the ? key at any time, you can display a list of these commands. For example, pressing the D key downloads a file. The H key brings up a help menu. To search the text of your current Web page, press the / key. This opens a line at the bottom of the screen where you enter your search pattern. Lynx then highlights the next instance of that pattern in the text. If you press N, Lynx displays the next instance. The \ key toggles you between a source and a rendered version of the current Web page, showing you the HTML tags or the formatted text.

Java for Linux: Blackdown

To develop Java applications, use Java tools, and run many Java products, you must install the Java 2 Software Development Kit (SDK) and the Java 2 Runtime Environment (JRE) on your system. Together they make up the Java 2 Platform, Standard Edition (J2SE). Sun currently supports and distributes Linux versions of these products. You can download them from Sun at **java.sun.com/j2se/1.3** and install them on your system. You can even select an RPM package version for easy installation on Red Hat. The current version of the J2SE is known as Java version 1.3. An earlier version, 1.2, will not work on Red Hat 7.0, though 1.3 will work on 7.1.

Though Sun supports Linux versions of Java, more thorough and effective Linux ports of Java can be obtained from the Blackdown project at **www.blackdown.org**. The Blackdown project has ported the J2SE, including versions 1.3 of the SDK and JRE. They have also ported previous versions of Java, including 1.1 and 1.2. More information and documentation are also available at this Blackdown Web site. The SDK and JRE 1.3 are usually available in the form of compressed archives, **.tar.bz2**. You use the **b2unzip** command to decompress the file and the **tar xvf** command to extract it. Extraction should be done in the **/usr/local** file. Follow the instructions in the **INSTALL** file to install the software.

Numerous additional Java-based products and tools are currently adaptable for Linux. Tools include Java 3D, Java Media Framework (JMF), and Java Advanced Imaging (JAI), all Blackdown projects (see Table 20-3). Many of the products run directly as provided by Sun. These include the HotJava Web browser and the Java Web server. You can download several directly from the Sun Java Web site at **java.sun.com**.

Table 20-3: Blackdown Java Packages and Java Web Applications

Application	**Description**
Java 2 Software Development Kit (SDK) 1.3	A Java development environment with a compiler, interpreters, debugger, and more. Part of the Java 2 Platform. Download the Linux port from **www.blackdown.org**.
Java 2 Runtime Environment 1.3	A Java Runtime Environment used to run Java applets. Part

Table 20-3: Blackdown Java Packages and Java Web Applications

Application	Description
(J2RE)	of the Java 2 Platform. Download the Linux port from **www.blackdown.org**.
Java 2 Platform SE (J2SE) for Linux	Java 2 Platform, Standard Edition, which includes Java 2 SDK and RE. Download the Linux port from **www.blackdown.org**.
Java 3D for Linux	Sun's 3D Application Program Interface for 3D Java programs. Download the Linux port from **www.blackdown.org**.
Java Media Framework (JMF) for Linux	Enable audio and video to be added to Java. Download the Linux port from **www.blackdown.org**.
Java Advanced Imaging (JAI) for Linux	Java Advanced Imaging API. Download the Linux port from **www.blackdown.org**.
Java 1.1 Development Kit (JDK) and Java 1.1 Runtime Environment (JRE)	The older Java 1.1 development environment with a compiler, interpreters, debugger, and more. Download the Linux port for your distribution's update through **www.blackdown.org**.
HotJava browser	Sun's HTML 3.2- and JDK 1.1-compliant Web browser. Download the Linux version from **java.sun.com**.
Java Web Server	A Web server implemented with Java. Available at Java Web site at **java.sun.com**.

Note See **java.sun.com/products** for an extensive listing of Java applications.

The Java 2 Software Development Kit: SDK

The Java Software Development Kit (SDK) provides tools for creating and debugging your own Java applets and provides support for Java applications, such as the HotJava browser. The kit includes demonstration applets with source code. You can obtain detailed documentation about the SDK from the Sun Web site at **java.sun.com**. Two major releases of the SDK are currently available-1.2, 1.3.*x*-with corresponding versions for the Java 2 Runtime Environment (J2RE) for 1.2 and 1.3. JAVA SDK adds capabilities for security, Swing, and running Java enhancements, such as Java3D and Java Sound.

Note Red Hat 7.1 still uses JRE 1.1, an earlier version of the J2RE that is compatible with older browsers.

SDK includes standard features found in the JDK features for internationalization, signed applets, JAR file format, AWT (window toolkit) enhancements, JavaBeans component model, networking enhancements, a math package for large numbers, database connectivity (JDBC), Object Serialization, and Inner Classes. Java applications include a Java compiler, javac; a Java debugger, jdb; and an applet viewer, appletviewer. In addition, the SDK offers the Java Naming and Directory Interface (JNDI), integrated Swing, Java 2d, network and security enhancements, and CORBA. With SDK, you can run the Blackdown port of Java 3D, Java Advanced Imaging, Java Media Framework, and Java Sound. Detailed descriptions of these features can be found in the SDK documentation.

Java Applets

You create a Java applet much as you would create a program using a standard programming language. You first use a text editor to create the source code, which is saved in a file with a **.java** extension. Then you can use the javac compiler to compile the source code file, generating a Java applet. This applet file has the extension **.class**. For example, the JDK demo directory includes the Java source code for a Blink applet called **Blink.java.** You can go to that directory and then compile the **Blink.java** file, generating a **Blink.class** file. The **example1.html** file in that directory runs the **Blink.class** applet. Start your browser and access this file to run the Blink applet.

```
# javac Blink.java
```

An applet is called within a Web page using the <applet> HTML tag. This tag can contain several attributes, one of which is required: code. You assign to code the name of the compiled applet. You can use several optional attributes to set features, such as the region used to display the applet and its alignment. You can even access applets on a remote Web site. In the following example, the applet called **Blink.class** is displayed in a box on the Web browser that has a height of 140 pixels and a width of 100 pixels, and is aligned in the center.

```
<applet code="Blink.class" width=100 height=140
align=center></applet>
```

To invoke the debugger, use the **appletviewer** command with the **-debug** option and the name of the HTML file that runs the applet.

```
appletviewer -debug mypage.html
```

Numerous Interface Development Environments (IDE) applications are available for composing Java applets and applications. Although most are commercial, some provide free shareware versions. An IDE provides a GUI interface for constructing Java applets. You can link to and download several IDE applications through the Blackdown Web page.

Web Search Utilities

To search for files on FTP sites, you can use search engines provided by Web sites, such as Yahoo!, Excite, Google, Alta Vista, or Lycos. These usually search for both Web pages and FTP files. To find a particular Web page you want on the Internet, you can use any number of online search sites such as Google, Yahoo!, Excite, Alta Vista, or Lycos. You can use their Web sites or perform searches from any number of Web portals, such as Netscape or Linux online. Web searches have become a standard service of most Web sites. Searches carried out on documents within a Web site may use local search indexes set up and maintained by indexing programs like ht:/Dig and WAIS. Sites using ht:/Dig use a standard Web page search interface, where WAIS provides its own specialized client programs like swais and xwais. WAIS is an older indexing program currently being supplanted by newer search applications like ht:/Dig. You can still obtain a free version of WAIS, called freeWAIS*,* from **ftp.cnidr.org**.

Creating Your Own Web Site

To create your own Web site, you need access to a Web server. Red Hat automatically installs the Apache Web server on its Linux systems. You can also rent Web page space on a remote server-a service many ISPs provide, some free. On Red Hat systems, the directory set up by your Apache Web server for your Web site pages is **/var/httpd/html**. Other servers provide you with a directory for your home page. Place the Web pages you create in that directory. You place your home page here. You can make other subdirectories with their own Web pages to which these can link. Web pages are not difficult to create. Links from one page to another move users through your Web site. You can even create links to Web pages or resources on other sites. Many excellent texts are available on Web page creation and management.

Web Page Composers

Web pages are created using HTML, the Hypertext Markup Language, which is a subset of Standard Generalized Markup Language (SGML). Creating an HTML document is a matter of inserting HTML tags in a text file. In this respect, creating a Web page is as simple as using a tag-based word processor. You use the HTML tags to format text for display as a Web page. The Web page itself is a text file you can create using any text editor, such as Vi. If you are familiar with tag-based word processing on Unix systems, you will find it conceptually similar to nroff. Some HTML tags indicate headings, lists, and paragraphs, as well as links to reference Web resources.

Instead of manually entering HTML code, you can use Web page composers. A Web page composer provides a graphical interface for constructing Web pages. The Linux version of WordPerfect can automatically generate a Web page from a WordPerfect document. You can create Web pages using all the word processing features of WordPerfect and Star Office. Special Web page creation programs, such as Netscape Composer, also can easily help you create complex Web pages without ever having to type any HTML tags explicitly. Remember, though, no matter what tool you use to create your Web page, the Web page itself will be an HTML document.

Note Many of the standard editors for the K Desktop and Gnome include Web page construction features. Many enable you to insert links or format headings. For example, the kedit program supports basic text-based Web page components. You can add headings, links, or lines, but not graphics.

Common Gateway Interfaces

A Common Gateway Interface (CGI) script is a program a Web server at a Web site can use to interact with Web browsers. When a browser displays a Web page at a particular Web site, the Web page may call up CGI programs to provide you with certain real- time information or to receive information from you. For example, a Web page may execute the server's **date** command to display the current date whenever the Web page is accessed.

A CGI script can be a Linux shell script, a Perl script, a Tcl/Tk program, or a program developed using a programming language such as C. Two special HTML operations are also considered CGI scripts: query text and forms. Both receive and process interactive responses from particular users. You have seen how a user can use a browser to display Web pages at a given Web site. In effect, the user is receiving information in the form of Web pages from the Web site. A user can also, to a limited extent, send information back to the Web site. This is usually information specifically prompted for in a Web page displayed by your browser. The Web server then receives and processes that information using the CGI programs.

A *form* is a Web page that holds several input fields of various types. These can be input boxes for entering text or check boxes and radio buttons that users simply click. The text boxes can be structured, allowing a certain number of characters to be entered, as in a phone number. They can also be unstructured, enabling users to type in sentences as they would for a comment. Forms are referred to as *form-based queries.* After entering information into a form, the user sends it back to the server by clicking a Submit button. The server receives the form and, along with it, instructions to run a specific CGI program to process the form.

Chapter 21: Network Tools

Overview

You can use a variety of network tools to perform tasks such as obtaining information about other systems on your network, accessing other systems, and communicating directly with other users. Network information can be obtained using utilities such as ping, finger, and host. Talk, ICQ, and IRC clients enable you to communicate directly with other users on your network. telnet performs a remote login to an account you may have on another system connected on your network. Each has a corresponding K Desktop or Gnome version. These provide a GUI interface, so you no longer have to use the shell command line to run these tools. In addition, your network may make use of older remote access commands. These are useful for smaller networks and enable you to access remote systems directly to copy files or execute commands.

Network Information: ping, finger, and host

You can use the **ping**, **finger**, **traceroute**, and **host** commands to find out status information about systems and users on your network. **ping** is used to check if a remote system is up and running. You use **finger** to find out information about other users on your network, seeing if they are logged in or if they have received mail. **host** displays address information about a system on your network, giving you a system's IP and domain name addresses. **traceroute** can be used to track the sequence of computer networks and systems your message passed through on its way to you. Table 21-1 lists various network information tools.

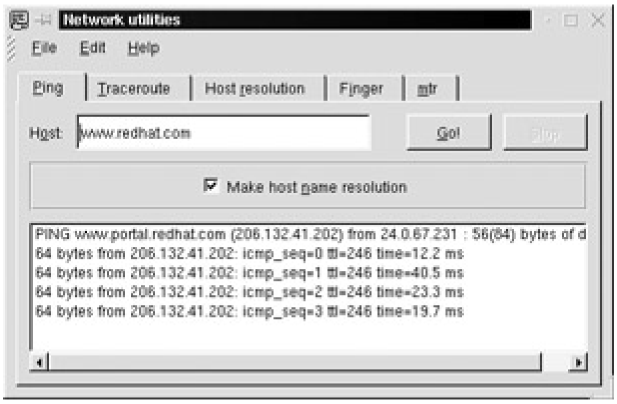
Table 21-1: Network Tools

Network Information Tool	Description
ping	Detects whether a system is connected to the network.
finger	Obtains information about users on the network.
who	Checks what users are currently online.
host	Obtains network address information about a remote host.
traceroute	Tracks the sequence of computer networks and hosts your message passes through.

Table 21-1: Network Tools

Network Information Tool	Description
knu	The KDE Network utilities featuring **finger**, **ping**, **traceroute**, and **host** commands.
Gnetutil	The Gnome Network utilities featuring **finger**, **ping**, **traceroute**, and **host** commands.
gfinger	Gnome finger client.
KFinger	Kde finger utility.
gHostLookup	Gnome utility to find IP addresses for hostnames.
Gwhois	Obtains information about networks, hosts, and users.

On the Gnome desktop, the Gnetutil utility provides a Gnome interface for entering the **ping**, **finger**, and **host** commands. On the K Desktop, you can use the KDE network utilities (knu) to issue **ping** , **finger**, **traceroute**, and **host** commands. Select the appropriate tabbed panel. For the Ping panel, enter the address of the remote system at the box labeled Host and click Go. The results are displayed in the pane below, as shown in Figure 21-1. Neither utility is installed on Red Hat 7.1, though you can obtain packages from **apps.kde.org** and **www.gnome.org**.



Host: www.redhat.com

☑ Make host name resolution

```
PING www.portal.redhat.com (206.132.41.202) from 24.0.67.231 : 56(84) bytes of d
64 bytes from 206.132.41.202: icmp_seq=0 ttl=246 time=12.2 ms
64 bytes from 206.132.41.202: icmp_seq=1 ttl=246 time=40.5 ms
64 bytes from 206.132.41.202: icmp_seq=2 ttl=246 time=23.3 ms
64 bytes from 206.132.41.202: icmp_seq=3 ttl=246 time=19.7 ms
```

Figure 21-1: K Desktop network utilities (knu) showing **ping**

ping

The **ping** command detects whether a system is up and running. **ping** takes as its argument the name of the system you want to check. If the system you want to check is down, **ping** issues a timeout message indicating a connection could not be made. The next example checks to see if **www.redhat.com** is up and connected to the network:

```
$ ping www.redhat.com
PING www.portal.redhat.com (206.132.41.231): 56 data bytes
64 bytes from 206.132.41.231: icmp_seq=0 ttl=248 time=24.0 ms
64 bytes from 206.132.41.231: icmp_seq=1 ttl=248 time=124.5 ms
64 bytes from 206.132.41.231: icmp_seq=2 ttl=248 time=77.9 ms
64 bytes from 206.132.41.231: icmp_seq=3 ttl=248 time=220.1 ms
64 bytes from 206.132.41.231: icmp_seq=4 ttl=248 time=14.9 ms

--- www.portal.redhat.com ping statistics ---
6 packets transmitted, 5 packets received, 16% packet loss
round-trip min/avg/max = 14.9/92.2/220.1 ms
```

You can also use **ping** with an IP address instead of a domain name. With an IP address, **ping** can try to detect the remote system directly without having to go through a domain name server to translate the domain name to an IP address. This can be helpful for situations where your network's domain name server may be temporarily down and you want to check if a particular remote host on your network is connected. In the next example, the user checks the Red Hat site using its IP address:

```
$ ping 206.132.41.231
PING 206.132.41.231 (206.132.41.231): 56 data bytes
64 bytes from 206.132.41.231: icmp_seq=0 ttl=248 time=16.6 ms
64 bytes from 206.132.41.231: icmp_seq=1 ttl=248 time=65.1 ms
64 bytes from 206.132.41.231: icmp_seq=2 ttl=248 time=70.1 ms
64 bytes from 206.132.41.231: icmp_seq=3 ttl=248 time=336.6 ms
64 bytes from 206.132.41.231: icmp_seq=4 ttl=248 time=53.6 ms
64 bytes from 206.132.41.231: icmp_seq=5 ttl=248 time=42.1 ms

--- 206.132.41.231 ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 16.6/97.3/336.6 ms
```

Note ping operation could also fail if ping access is denied by a network's firewall. See Chapter 39 for more details.

finger and who

You can use the **finger** command to obtain information about other users on your network and the **who** command to see what users are currently online on your system. The **who** command lists all users currently connected along with when, how long, and where they logged in. It has several options for specifying the level of detail. **who** is meant to operate on a local system or network. **finger** operates on large networks, including the Internet. **finger** checks to see when a user last logged in, the type of shell that he or she is using, the pathname of the home directory, and whether any mail has been received. **finger** then checks for a **.plan** file in a user's home directory that may contain information about the user. The **.plan** file is a file you create yourself on your own home directory. You can place information you want made publicly available into the **.plan** file. You can enter the command **finger** on the command line with the login name of the user you want to check.

On the Gnome desktop you can use the gfinger utility to issue finger commands, and in the K Desktop you can use the KDE network utilities (knu). Click the Finger panel and enter the address of the host you want to check. On the K Desktop, the KFinger tool also provides a GUI for easily sending finger queries. It features entries for users and remote servers. You can search for users on specific remote systems.

host

With the **host** command, you can find network address information about a remote system connected to your network. This information usually consists of a system's IP address, domain name address, domain name nicknames, and mail server. This information is obtained from your network's domain name server. For the Internet, this includes all systems you can connect to over the Internet.

The **host** command is an effective way to determine a remote site's IP address or URL. If you have only the IP address of a site, you can use **host** to find out its domain name. For network administration, an IP address can be helpful for making your own domain name entries in your **/etc/host** file. That way, you needn't rely on a remote domain name server (DNS) for locating a site. On the K Desktop, you can use the KDE network utilities for running host commands. Click the Host resolution panel and enter the address of the host you want to check. On Gnome, you can use the gHostLookup utility.

```
$ host www.gnome.org
www.gnome.org is a nickname for gnome.labs.redhat.com
gnome.labs.redhat.com has address 199.183.24.235
gnome.labs.redhat.com mail is handled (pri=10) by mail.redhat.com

$ host 199.183.24.235
235.24.183.199.IN-ADDR.ARPA domain name pointer
gnome.labs.redhat.com
```

The Gwhois program is a Gnome-based client that displays information about hosts obtained from NIC network services. Gwhois provides an X Windows interface with a list of NIC servers from which to choose. gHostLookup is a simple Gnome application that returns a machine's IP address when you give it the hostname.

Part V: Red Hat Servers

Chapter List

Chapter 22: Server Management

Overview

Reflecting the close relationship between Unix and the development of the Internet, Linux is particularly good at providing Internet services, such as the Web, FTP, and e-mail. In the case of the Web, instead of only accessing other sites, you can set up your own Linux system as a Web site. Other people can then access your system using Web pages you created or download files you provide for them. A system that operates this way is called a *server* and is known by the service it provides. You can set up your system to be a Web server or an FTP server, connecting it to the Internet and turning it into a site others can access. A single Linux system can provide several different services. Your Linux system can be a Web server and an FTP server, as well as a mail and news server, all at the same time. One user could download files using your FTP services, while another reads your Web pages. All you have to do is install and run the appropriate server software for each service. Each one operates as a continually running daemon looking for requests for its particular services from remote users. A *daemon* is any program that continually runs, checking for certain requests and performing appropriate actions.

When you install Linux, you have the option of installing several Internet servers, including Web and FTP servers. Linux was designed with Internet servers in mind. For many Linux distributions, a standard install installs these servers automatically and configures them for you (check to make sure the servers are included). Every time you start your system, you also start the Web and FTP server daemons. Then, to turn your Linux system into a Web server, all you have to do is create Web pages. For an FTP server, you only have to place the files you want to make available in the FTP directories.

You can operate your Linux system as a server on the Internet or an intranet (local area network), or you can set it up to service only the users on your own system. To operate servers as Internet servers, you must obtain a connection to the Internet and provide access to your system for remote users. Access is usually a matter of enabling anonymous logins to directories reserved for server resources. Linux systems are usually already configured to enable such access for Web and FTP users. Connections to the Internet that can accommodate server activity can be difficult to find. You may need a dedicated connection, or you may need to use a connection set up by an Internet service provider (ISP). You are no longer connecting only yourself to the Internet, but you are allowing many other users to make what could be a great many connections to you through the Internet. This will involve security risks to your system, and precautions should be taken to protect it (see Chapter 39). If you only want to provide the services to a local area network (LAN), you don't need a special connection. Also, you can provide these services to users by allowing them to connect over a modem and to log in directly. Users could dial into your system and use your Web pages or use FTP to download files. Furthermore, users with accounts on your own machine can also make use of the servers. In whatever situation you want to use these services, you need the appropriate server software installed and running. This chapter examines how servers are started and stopped on your system, as well as different ways of accessing the servers.

Note Linuxconf provides added modules on its Web site that you can use to configure most Internet servers, including the Apache Web server, the BIND domain name server, the Washington University FTP server, and the Sendmail mail server.

Starting Servers: Standalone and xinetd

A *server* is a daemon that runs concurrently with your other programs, continuously looking for a request for its services, either from other users on your system or from remote users connecting to your system through a network. When it receives a request from a user, a server starts up a session to provide its services. For example, if users want to download a file from your system, they can use their own FTP client to request that your FTP server start a session for them. In the session, they can access and download files from your system. Your server needs to be running for a user to access its services. For example, if you set up a Web site on your system with HTML files, you must have the **httpd** Web server program running before users can access your Web site and display those files. See Chapters 23 and 24 on how to install FTP and Web servers.

You can start a server in several ways. One way is to do it manually from the command line by entering the name of the server program and its arguments. When you press ENTER, the server starts, although your command line prompt reappears. The server runs concurrently as you perform other tasks. To see if your server is running, you can enter the following command to list all currently running processes. You should see a process for the server program you started. To refine the list, you can add a **grep** operation with a pattern for the server name you want. The second command lists the process for the Web server.

```
# ps -aux
# ps -aux | grep 'httpd'
```

On Red Hat Linux systems, you use special startup scripts to start and stop your server manually. These scripts are located in the **/etc/rc.d/init.d** directory and have the same name as the server programs. For example, the **/etc/rc.d/init.d/httpd** script with the **start** option starts the Web server. Using this script with the **stop** option stops it. Instead of using the complete pathname for the script, you can use the service command and the script name. The following commands are equivalent.

```
/etc/rc.d/init.d/httpd stop
service httpd stop/
```

Instead of manually executing all the server programs each time you boot your system, you can have your system automatically start the servers for you. You can do this in two ways, depending on how you want to use a server. You can have a server running continuously from the time you start your system until you shut it down, or you can have the server start only when it receives a request from a user for its services. If a server is being used frequently, you may want to have it running all the time. If it is used rarely, you may only want the server to start when it receives a request. For example, if you are running a Web site, your Web server is receiving requests all the time from remote hosts on the Internet. For an FTP site, however, you may receive requests infrequently, in which case you may want to have the FTP server start only when it receives a request. Of course, certain FTP sites receive frequent requests, which would warrant a continuously running FTP server.

A server that starts automatically and runs continuously is referred to as a *standalone* server. Red Hat uses the SysV Init procedure to start servers automatically whenever your system boots. This procedure uses special startup scripts for the servers located in the **/etc/rc.d/init.d** directory. Most Linux systems configure the Web server to start automatically and to run continuously by default. A script for it called **httpd** is in the **/etc/rc.d/init.d** directory.

To start the server only when a request for its services is received, you configure it using the **xinetd** daemon. If you add, change, or delete server entries in the **/etc/xinetd** files, you will have to restart the **xinetd** daemon for these changes to take effect. On Red Hat, you can restart the **xinetd** daemon using the **/etc/rc.d/init.d/xinetd** script with the **restart** argument, as shown here:

```
# /etc/rc.d/init.d/xinetd restart
```

You can also use the **xinetd** script to start and stop the **xinetd** daemon. Stopping effectively shuts down all the servers that the **xinetd** daemon manages (those listed in the **/etc/xinetd.conf** file or **xinetd.d** directory).

```
# service xinetd stop
# service xinetd start
```

You can also directly restart the **xinetd** by sending its process a SIGHP signal, forcing it to restart. To do this, you use the **kill** command with the **-HUP** option and the process ID of the

xinetd daemon. You can find the process ID using the **ps -aux** command to list all processes and then use **grep** to locate the **xinetd** entry as shown here. The process ID will also be held in the /**var/run/xinetd.pid** file.

```
# ps -aux | grep sinetd
# kill -HUP xinetd-process-id
```

Note Versions prior to 7.0 and other Linux systems used the **inetd** daemon (the term stands for the Internet Services Daemon) instead of **xinetd. xinetd** is meant to be the enhanced replacement for **inetd**. If you are upgrading from **inetd**, you can use the **inetdconvert** command to convert **inetd** entries into **xinetd** configurations.

Service Management Tools: ntsysv and serviceconf, chkconfig, and System V Init

On Red Hat, the System V Init Editor, the Red Hat Setup ntsysv utility, and the **ch**kconfig command provide simple interfaces you can use to choose what servers you want started up and how you want them to run. You use these tools to control any daemon you want started up, including system services such as cron, the print server, and remote file servers for Samba and NFS; authentication servers for Kerberos; and, of course, Internet servers for FTP or HTTP. Such daemons are referred to as *services*, and you should think of these tools as managing these services. Any of these services can be set up to start or start at different runlevels.

These tools manage services that are started up by scripts in the **/etc/rc.d/init.d** directory. If you add a new service, both the chkconfig and System V Init Editor can manage it. As described in the following section, services are started up at specific runlevels using startup links in various runlevel directories. These links are connected to the startup scripts in the **init.d** directory. Runlevel directories are numbered from 0 to 6 in the **/etc/rc.d** directory (e.g., **/etc/rc.d/rc3.d** for runlevel 3 and **/etc/rc.d/rc5.d** for runlevel 5). Removing a service from a runlevel only removes its link in the corresponding runlevel **rc.d** directory. It does not touch the startup script in the **init.d** directory. Having a server start at a specified runlevel puts the link back in that runlevel directory. For example, if you specify that **httpd** is no longer to start at runlevel 3, then the **S85httpd** startup link in the **rc3.d** directory is deleted. Having **httpd** start at runlevel 5 re-creates the **S85htppd** link in the **rc5.d** directory. See the later section on SysV Init scripts for more details, and Chapter 28 for more information on runlevels.

Note You can also control the startup of a server using Linuxconf or Webmin. On Linuxconf, the Control Service Activity panel, located in the Control Panel list, lists various services available on your system. On Webmin, the Bootup and Shutdown page, accessed from the System page, lists the daemons and servers that you can have automatically start when the system boots.

ntsysv

With the Red Hat ntsysv and serviceconf utilities, you can simply select from a list of commonly used services you want to run when your system boots up (see Figure 22-1). The utility serviceconf also lets you start, stop, and restart a server, much like the service command. You can also set startup runlevels, just as you can with chkconfig. Both netsysv and serviceconf will display a list of your installed servers. You can access ntsysv from the Text Mode Setup menu by selecting Services (see Chapter 30). It can run on any command line interface. You can access serviceconf from the System Settings window within the Start Here window on the Gnome desktop. It will be labeled Service Configuration.

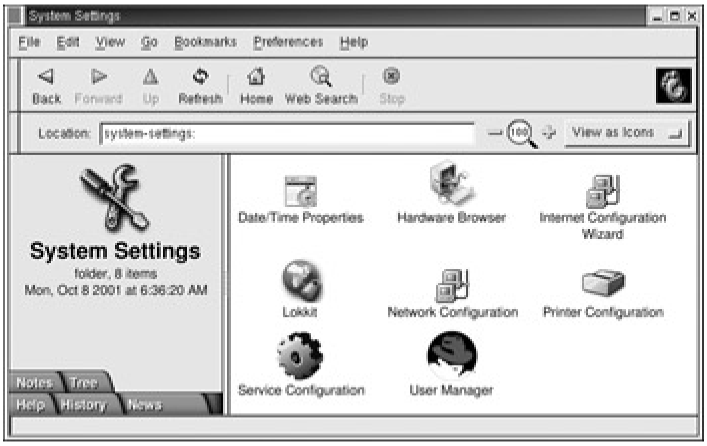


Figure 22-1: Service Configuration (serviceconf)

chkconfig

With the **chkconfig** command you can specify the service you want start and the level you want to start it at. Unlike other service management tools, chkconfig works equally well on standalone and xinetd services. Although standalone services can be run at any runlevel, you can also turn xinetd services on or off for the runlevels that xinetd runs in. Table 22-1 lists the different **chkconfig** options.

You use the **on** option to have a service started at certain runlevels, and the **off** option to not have it started. You can specify the runlevel to use with the **--level** option. If no level is specified, then chkconfig will use any chkconfig default information in a service's **init.d** startup script. Red Hat installs its services with chkconfig default information already entered (should this be missing, chkconfig will use runlevels 3, 4, and 5). The following example will have the Web server (httpd) started at runlevel 5.

```
chkconfig --level 5 httpd on
```

The **off** option will actually configure a service to shut down if it enters a specified runlevel. This example will shut down the Web sever if runlevel 3 is entered. If it is not running, it remains shut down:

```
chkconfig --level 3 httpd off
```

Table 22-1: chkconfig Options

Option	**Description**
--level *runlevel*	Specifies a runlevel to turn on, off, or reset a service.
--list *service*	Lists startup information for services at different runlevels. xinetd services are just on or off. With no argument, all services are listed, including xinetd services.
--add *service*	Adds a service, creating links in default specified runlevels (or all if none specified).
--del *service*	Deletes all links for the service (startup and shutdown) in all runlevel directories.

Table 22-1: chkconfig Options

Option	Description
service **on**	Turns a service on, creating a startup link in the specified or default runlevel directories.
service **off**	Turns a service off, creating shutdown links in specified or default directories
service **reset**	Resets a service startup information, creating default links as specified in the chkconfig entry in the service's **init.d** startup script.

The **reset** option will restore a service to its chkconfig default options as specified in the service's **init.d** startup script:

```
chkconfig wu-ftpd reset
```

To see just the startup information for a service, you use just the service name with the **--list** option:

```
chkconfig --list httpd
httpd   0:off  1:off  2:off     3:on     4:off     5:on   6:off
```

Unlike the System V Init Editor and ntsysv, chkconfig also has the ability to have xinetd services enabled or disabled. Simply enter the xinetd service with either an **on** or **off** option. The service will be started up or shut down and its xinetd configuration script in the **/etc/xinetd.d** directory will have its disable line edited accordingly. For example, to start swat, the Samba configuration server, which runs on xinetd, you simply enter:

```
chkconfig swat on
chkconfig --list swat
     swat          on
```

The swat configuration file for xinetd, **/etc/xinetd.d/swat**, will have its disable line edited to "no," as shown here:

```
disable=no
```

Should you want to shut down the swat server, you can use the **off** option. This will change the disable line in **/etc/xinetd.d/swat** to read disable=yes.

```
chkconfig swat off
```

The same procedure works for other xinetd services such as wu-ftpd, the FTP server, and finger.

Should you want a service removed entirely from the entire startup and shutdown process in all runlevels, you can use the **--del** option. This removes all startup and shutdown links in all the runlevel directories.

```
chkconfig --del httpd
```

You can also have services added to management by chkconfig with the **--add** option. chkconfig will create startup links for it in the appropriate startup directories, **/etc/rc.d/rc*n*.d**. If you have previously removed all links for a service, you can restore them with the **--add** option:

```
chkconfig --add httpd
```

To see a list of services managed by https, you use the **--list** option. A sampling of services managed by chkconfig are shown here. The on and off status of the service is shown at each runlevel. xinted services and their status are also shown:

```
chkconfig -list
dhcpd               0:off     1:off     2:off     3:off     4:off     5:off
6:off
httpd               0:off     1:off     2:off     3:off     4:off     5:off
6:off
named               0:off     1:off     2:off     3:off     4:off     5:off
6:off
tux                 0:off     1:off     2:off     3:off     4:off     5:off
6:off
kudzu               0:off     1:off     2:off     3:on     4:on     5:on
6:off
innd                0:off     1:off     2:off     3:off     4:off     5:off
6:off
lpd                 0:off     1:off     2:on     3:on     4:on     5:on
6:off
nfs                 0:off     1:off     2:off     3:off     4:off     5:off
6:off
smb                 0:off     1:off     2:off     3:off     4:off     5:off
6:off
crond               0:off     1:off     2:on     3:on     4:on     5:on
6:off
xinetd              0:off     1:off     2:off     3:on     4:on     5:on
6:off
xinetd based services:
     time:     off
     finger:     off
     pop3s:     off
     swat:     on
     wu-ftpd:     off
```

chkconfig works by creating startup and shutdown links in the appropriate runlevel directories in the **/etc/rc.d** directory. For example, when chkconfig added the httpd service at runlevel 5, it created a link in the **/etc/rc.d/rc5.d** directory to the startup script **httpd** in the **/etc/rc.d/init.d** directory. When it turned off the Web service from runlevel 3, it created a shutdown link in the **/etc/rc.d/rc3.d** directory to use the script **httpd** in the **/etc/rc.d/initd** directory to make sure the Web service is not started. In the following example, the user turns on the Web service (httpd) on runlevel 3, creating the startup link in rc5.d, S85httpd, and then turns off the Web service on runlevel 3, creating a shutdown link in rc3.d, K15httpd.

```
chkconfig --level 5 httpd on
ls /etc/rc.d/rc3.d/*httpd
   /etc/rc.d/rc3.d/K15httpd
chkconfig -level 3 httpd off
ls /etc/rc.d/rc3.d/*httpd
   /etc/rc.d/rc3.d/K15httpd
```

Default runlevel information should be placed in the startup scripts that are to be managed by chkconfig. Red Hat has already placed this information in the startup scripts for the services that are installed with its distribution. You can edit these scripts to change the default information if you wish. This information is entered as a line beginning with a # sign and followed by the chkconfig keyword and a colon. Then you list the default runlevels that the service should start up on, along with the start and stop priorities. The following entry lists runlevels 3 and 5 with a start priority of 85 and a stop of 15:

```
# chkconfig: 35 85 15
```

Now when a user turns on the httpd service with no level option specified, chkconfig will start up httpd at runlevels 3, 4, and 5:

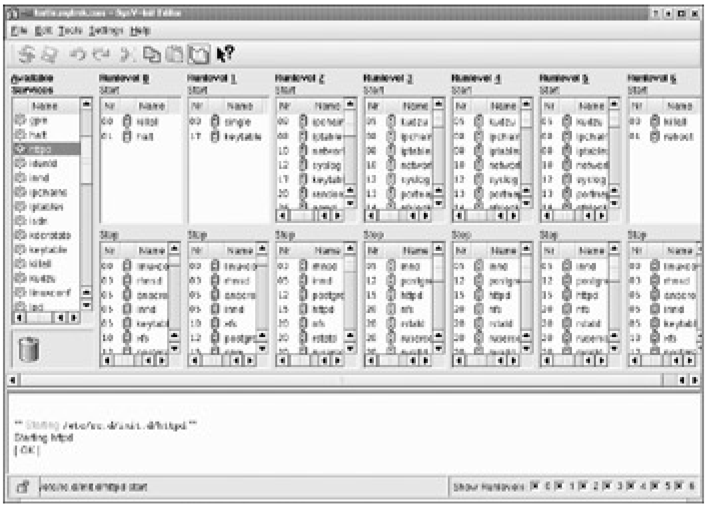
```
chkconfig httpd on
```

A description line should also be added that chkconfig can use to describe the service. With the description you enter a short description of the service, using the \ symbol before a newline to use more than one line:

```
# description: Apache is a World Wide Web server. It is used to serve \
# HTML files and CGI.
```

System V Init Editor

The System V Init Editor is accessible on Red Hat from the KDE desktop's system menu. It features a GUI interface to enable you to manage any daemons on your system easily-Internet servers as well as system daemons, such as print servers. The Init Editor window is divided into three major panes. To the left is a scroll window labeled "available" that lists all the daemons available for use on your system. These include the daemons for Internet servers, such as **httpd**. To the right, taking up most of the window, are an upper pane and a lower pane. The upper pane has scroll windows, one for each runlevel. These list the daemons currently configured to run in their respective runlevels. The lower pane also holds scroll windows, one for each runlevel. These are daemons that will be shut down if you switch to that respective runlevel. System administrators can switch from one level to another. All the servers that start up under normal processing are listed in the start runlevel 5 (graphical login) and runlevel 3 (command line) scroll windows. Figure 22-2 shows the System V Init Editor.



```
** Starting /etc/rc.d/init.d/httpd **
Starting httpd
[ OK ]
```

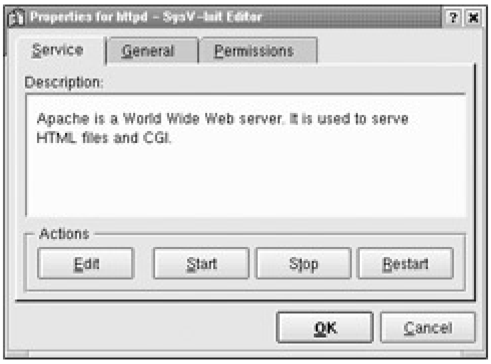
/etc/rc.d/init.d/httpd start

Figure 22-2: The System V Init Editor

Note The publisher's edition of Red Hat Linux does not include KDE System V Init Editor. Instead you can run an older version called System V Runlevel Editor, accessible from the Control Panel. It operates much the same as the System V Init Editor, with few differences.

You can easily configure a server to start automatically when you boot at a certain runlevel. Your system operates at certain runlevels, each specified by a given number, such as the standard multiuser level (runlevel 3), a graphical login level (runlevel 5), and an administrative level (runlevel 1). See Chapter 28 for a discussion of runlevels. To have a server start at a given runlevel, click and drag its entry in the available scroll window to the scroll window for the runlevel you want it to start at. For example, if you use a graphical login, your runlevel when you start up is 5. To have the Web server, **httpd**, start up automatically whenever you boot your system, check to see if it is in the runlevel 5 startup scroll window. If not, then click and drag the **httpd** entry in the available window to the runlevel 5 startup window. Select a position in that window where there appear to be available numbers, such as between 15 or 17, or at the end of the scroll window. An **httpd** entry will then appear in the scroll window. If you do not want to have the service started up automatically, you can remove it from its startup scroll window by clicking on it and selecting Cut from the pop-up menu. So, to remove **httpd** from the runlevel 5 startup window, click on its entry in that window and select Cut.

To start or stop a server manually, click its entry in the available scroll window. A Properties window appears where you can select whether you want to start, stop, or restart the server. Do the same procedure to start the server, clicking the Start button. An Edit button will open the server's startup script in an editor and let you modify it directly. Figure 22-3 shows the System V Init Editor **httpd** Properties windows.



Description:

Apache is a World Wide Web server. It is used to serve HTML files and CGI.

Figure 22-3: The System V Init Editor httpd Properties window

Servers that operate under **xinetd** are not listed by the System V Init Editor. The FTP **wu-ftpd** server is usually installed to run under **xinetd**, so you won't find entries for them here. The System V Editor reads its list of servers from the server scripts in the **/etc/rc.d/init.d** directory. If you add a new script, you can have the System V Editor rescan that directory and then see it appear in the available list. Removing a server from a runlevel window only removes its link in the corresponding runlevel **rc.d** directory. It does not touch the startup script in the **init.d** directory. Adding in the server to the start runlevel window puts the link back in that runlevel directory. Adding a server to a stop window adds a *K* link in the corresponding **rc.d** directory, which stops a server when the system switches to that runlevel. For example, if you remove **httpd** from the runlevel 3 start window, then the **S85httpd** link in the **rc3.d** directory is deleted. Adding **httpd** back to the runlevel 3 start window re-creates the **S85htppd** link in the **rc3.d** directory. Adding **httpd** to the runlevel 2 stop directory would create a **K85httpd** link in the **rc2.d** directory, shutting down the server when switching to runlevel 2.

SysV Init: init.d Scripts

The startup and shutdown of server daemons is managed using special startup scripts located in the **/etc/rc.d/init.d** directory. These scripts often have the same name as the server's program. For example, for the **/usr/sbin/httpd** Web server program, a corresponding script is called **/etc/rc.d/init.d/httpd**. This script actually starts and stops the Web server. This method of using **init.d** startup scripts to start servers is called *SysV Init,* after the method used in UNIX System V.

Note If you change the configuration of a server, you may need to start and stop it several times as you refine the configuration. Several servers provide special management tools that enable you to perform this task easily. The apachectl utility enables you to start and stop the Apache Web server easily. It is functionally equivalent to using the **/tec/rc.d/init.d/httpd** script to start and stop the server. For the domain name server, the ndc utility enables you to start and stop the named server. It is, however, advisable not to mix the use of **init.d** scripts and the management tools.

The startup scripts in the **/etc/rc.d/init.d** directory can be executed automatically whenever you boot your system. Be careful when accessing these scripts, however. These start essential programs, such as your network interface and your printer daemon. These init scripts are accessed from links in subdirectories set up for each possible runlevel. In the **/etc/rc.d** directory is a set of subdirectories whose names have the format **rc*N*.d**, where *N* is a number referring to a runlevel. The **rc** script detects the runlevel in which the system was started, and then executes only the startup scripts specified in the subdirectory for that runlevel. The two runlevels most commonly used are 3, the multiuser level, and 5, the graphical login. When you start your system, the **rc** script executes the startup scripts specified in the **rc3.d** directory, if you are performing a command line login, and **rc5.d** if you are using a graphical login. The **rc3.d** and **rc5.d** directories hold symbolic links to certain startup scripts in the **/etc/rc.d/init.d** directory. So, the **httpd** script in the **/etc/rc.d/init.d** directory is actually called through a symbolic link in the **rc3.d** or the **rc5.d** directory. The symbolic link for the **/etc/rc.d/httpd** script in the **rc3.d** directory is **S85httpd**. The *S* prefixing the link stands for "startup" and calls the corresponding **init.d** script with the **start** option. The number indicates the order in which startup scripts are run, lower ones first. **S85httpd** invokes **/etc/rc.d/init.d/httpd** with the option **start**. The numbers in these links are simply there for ordering purposes. If you change the name of the link to start with a *K,* then the script is invoked with the **stop** option, stopping it. Such links are used in the runlevels 0 and 6 directories, **rc6.d** and **rc0.d**. Runlevel 0 halts the system and runlevel 6 reboots it. You can use the **runlevel** command to find out what runlevel you are currently operating at (see Chapter 28 for more details on runlevels). A listing of runlevels is shown here:

Runlevel	**rc.d Directory**	**Description**
0	rc0.d	Halt (shut down) the system
1	rc1.d	Single-user mode (no networking, limited capabilities)
2	rc2.d	Multiuser mode with no NFS support (limited capabilities)
3	rc3.d	Multiuser mode (full operational mode)
5	rc5.d	Multiuser mode with graphical login (full operation mode with graphical login added)
6	rc6.d	Reboot system

Most server software using RPM Red Hat packages will automatically install the startup scripts and create the needed links in the appropriate **rc*N*.d** directories. Startup scripts, though, can be used for any program you may want run when your system starts up. To have such a program start automatically, you first create a startup script for it in the **/etc/rc.d/init.d** directory, and then create symbolic links to that script in the **/etc/rc.d/rc3.d** and **/etc/rc.d/rc5.d** directories. A shutdown link (*K*) should also be placed in the **rc6.d** directory used for runlevel 6 (reboot).

A simplified version of the startup script **httpd** used on Red Hat systems is shown here. You can see the different options listed under the case statement: start, stop, status, restart, and reload. If no option is provided (*), then the script use syntax is displayed. The **httpd** script first executes a script to define functions used in these startup scripts. The **daemon** function with **httpd** actually executes the **/usr/sbin/httpd** server program.

```
echo -n "Starting httpd: "
 daemon httpd
 echo
 touch /var/lock/subsys/httpd
```

The **killproc** function shuts down the daemon. The lock file and the process ID file (**httpd.pid**) are then deleted.

```
 killproc httpd
 echo
 rm -f /var/lock/subsys/httpd
 rm -f /var/run/httpd.pid
```

The **daemon**, **killproc**, and **status** scripts are shell scripts defined in the **functions** script also located in the **inet.d** directory. The **functions** script is executed at the beginning of each startup script to activate these functions. A list of these functions is provided in Table 22-2.

```
. /etc/rc.d/init.d/functions
```

The beginning of the startup script holds tags used to configure the server. These tags, which begin with an initial #, are used to provide runtime information about the service to your system. The tags are listed in Table 22-2 along with the startup functions. You enter a tag with a preceding # symbol, the tag name with a colon, and then the tag arguments. For example, the processname tag will specify the name of the program being executed, in this example **httpd**.

Table 22-2: System V init Script Functions and Tags

Init Script Function	**Description**
daemon [+/-*nicelevel*] *program* [*arguments*] [&]	Starts a daemon, if it is not already running.
killproc *program* [*signal*]	Sends a signal to the program; by default it sends a SIGTERM, and if the process doesn't stop, it sends a SIGKILL. It will also remove any PID files, if it can.
pidofproc *program*	Used by another function, it determines the PID of a program.
status *program*	Displays status information.
Init Script Tag	**Description**
# chkconfig: *startlevellist startpriority endpriority*	Required. Specifies the default start levels for this service as well as start and end priorities.
# description [*ln*]**:** *description of service*	Required. The description of the service, continued with '\' characters. Use an initial # for any added lines. With the *ln* option, you can specify the language the description is written in.
# autoreload: true	Optional. If this line exists, the daemon checks its configuration files and reloads them automatically when they change.
# processname: *program*	Optional, multiple entries allowed. Name of the program or daemon started in the script.
# config: *configuration-file*	Optional, multiple entries allowed. Specify a configuration file used by the server.
# pidfile: *pid-file*	Optional, multiple entries allowed. Specifies the PID file.
# probe: true	Optional, used *in place* of autoreload, processname, config, and pidfile entries to automatically probe and start the service.

```
# processname: httpd
```

If your script starts more than one daemon, you should have a processname entry for each. For example, the Samba service starts up both the **smdb** and **nmdb** daemons.

```
# processname: smdb
# processname: nmdb
```

The end of the tag section is indicated by an empty line. After this line, any lines beginning with a # are treated as comments. The chkconfig line will list the default runlevels that the service should start up on, along with the start and stop priorities. The following entry lists runlevels 3, 4, and 5 with a start priority of 85 and a stop of 15.

```
# chkconfig: 345 85 15
```

With the description you enter a short description of the service, using the \ symbol before a newline to use more than one line. pidfile indicates the file where the server's process ID is held. With config tags, you specify the configuration files the server may use. In the case of the Apache Web server, there may be three configuration files:

```
# config: /etc/httpd/conf/access.conf
# config: /etc/httpd/conf/httpd.conf
# config: /etc/httpd/conf/srm.conf
```

As an example, a simplified version of the Web server startup script is shown here. Most scripts are much more complicated, particularly when determining any arguments or variables a server may need to specify when it starts up. It has the same name as the Web server daemon, **httpd**.

/etc/rc.d/init.d/httpd

```
#!/bin/sh
#
# Startup script for the Apache Web Server
#
# chkconfig: 35 85 15
# description: Apache is a World Wide Web server. It is used to serve \
# HTML files and CGI.
# processname: httpd
# pidfile: /var/run/httpd.pid
# config: /etc/httpd/conf/access.conf
# config: /etc/httpd/conf/httpd.conf
# config: /etc/httpd/conf/srm.conf


# Source function library.
. /etc/rc.d/init.d/functions

# See how we were called.
case "$1" in
      start)
            echo -n "Starting httpd: "
             daemon httpd
             echo
             touch /var/lock/subsys/httpd
              ;;
     stop)
             killproc httpd
             echo
             rm -f /var/lock/subsys/httpd
             rm -f /var/run/httpd.pid
             ;;
    status)
             status httpd
             ;;
    restart)
             $0 stop
             $0 start
             ;;
    reload)
             echo -n "Reloading httpd: "
             killproc httpd -HUP
             echo
             ;;
    *)
             echo "Usage: $0 {start|stop|restart|reload|status}"
             exit 1
   esac

exit 0
```

The RPM packaged versions for an Internet server include the startup script for that server. Installing the RPM package installs the script in the **/etc/rc.d/init.d** directory and creates its appropriate links in the runlevel directories, such as **/etc/rc.h/rc3.d**. If you decide, instead, to create the server using its source code files, you can then manually install the startup script. If no startup script exists, you first make a copy of the **httpd** script-renaming it-and then edit the copy to replace all references to **httpd** with the name of the server daemon program. Then, place the copy of the script in the **/etc/rc.d/ init.d** directory and make a symbolic link to it the **/etc/rc.d/rc3.d** directory. Or you could use the System V Init Editor to create the link in the **/etc/rc.d/rc3.d** directory. Have the editor scan the **init.d** directory by selecting Re-scan from the File menu, click the entry in the Available listing, click the Add button, and then select the Runlevel and Start options in the Add window. When you start your system now, the new server is automatically started up, running concurrently and waiting for requests.

Extended Internet Services Daemon (xinetd)

If your system averages only a few requests for a specific service, you don't need the server for that service running all the time. You only need it when a remote user is accessing its service. The Extended Internet Services Daemon (**xinetd)** manages Internet servers, invoking them only when your system receives a request for their services. **xinetd** checks continuously for any requests by remote users for a particular Internet service; when it receives a request, it then starts the appropriate server daemon.

The **xinetd** program is designed to be a replacement for **inetd**, providing security enhancements, logging support, and even user notifications. For example, with **xinetd** you can send banner notices to users when they are not able to access a service, telling them why. **xinetd** security capabilities can be used to prevent denial-of-service attacks, limiting remote hosts' simultaneous connections or restricting the rate of incoming connections. **xinetd** also incorporates TCP, providing TCP security without the need to invoke the **tcpd** daemon.

Furthermore, you do not have to have a service listed in the **/etc/services** file. **xinetd** can be set up to start any kind of special-purpose server. The Red Hat Linux versions 7.0 and up use **xinetd**. Many older Linux systems may still be using **inetd**.

You can start, stop, and restart **xinetd** using its startup script in the **/etc/rc.d/init.d** directory, as shown here:

```
# /etc/rc.d/init.d/xinetd stop
# /etc/rc.d/init.d/xinetd start
# /etc/rc.d/init.d/xinetd restart
```

On Red Hat, you can also turn on and off particular xinetd services with chkconfig, as described earlier. Use the on and off options to enable or disable a service. chkconfig will edit the disable option for the service, changing its value to "yes" for off and "no" for on. For example, to enable the swat server, you could enter:

```
chkconfig swat on
```

The **xinetd.conf** file is the configuration file for **xinetd**. Entries in it define different servers to be activated when requested along with any options and security precautions. An entry consists of a block of attributes defined for different features, such as the name of the server program, the protocol used, and security restrictions. Each block for an Internet service such as a server is preceded by the keyword **service** and the name by which you want to identify the service. A pair of braces encloses the block of attributes. Each attribute entry begins with the attribute name followed by an assignment operator such as = and then the value or values assigned. A special block specified by the keyword **default** contains default attributes for services. The syntax is shown here:

```
service <service_name>
{
<attribute> <assign_op> <value> <value> ...
 ...
}
```

Most attributes take a single value for which you use the standard assignment operator, =. Some attributes can take a list of values. You can assign values with the = operator, but you can also add or remove items from these lists with the =+ and =**-** operators. Use the =+ to add values and =**-** to remove values. You often use the =+ and =**-** operators to add values to attributes that may have an initial value assigned in the default block.

Attributes are listed in Table 22-3. Certain attributes are required for a service. These include **socket_type** and **wait**. **socket_type** can be *stream* for stream-based service, *dgram* for datagram-based service, *raw* for service that requires direct access, and *seqpacket* service for sequential datagram transmission. The wait attribute can have a yes or no value to specify if the server is single-threaded (yes) or multithreaded (no). If yes, then **xinetd** will wait, calling the server initially and letting that server handle further requests until the server stops. If the value is no (multithreaded), **xinetd** will continue to handle new requests for the server, generating new server processes to handle them. For a standard Internet service, you would also need to provide the user (user ID for the service), the server (name of the server program), and the protocol (protocol used by the server). With **server_arguments**, you can also list any arguments you want passed to the server program (this does not include the server name as with **tcpd**). If protocol is not defined, the default protocol for the service is used.

```
service ftp
{
 socket_type = stream
 wait = no
 user = root
 protocol = ftp
 server = /usr/sbin/in.ftpd
 server_args = -l -a
 disable = yes
}
```

Services can be turned on and off with the **disable** attribute. In the previous example, the **disable** attribute has turned off the FTP service, keeping the FTP server shut down. To enable a service, you would set the **disable** attribute to **no**, as shown here:

```
 disable = no
```

You then have to restart **xinetd** to start the service.

```
# /etc/rc.d/init.d/xinetd restart
```

Note Red Hat currently disables all the services it initially set up when it installed **xinetd**. To enable a particular service you will have to set its **disable** attribute to no.

To enable management by chkconfig, a commented default and description entry need to placed before each service segment. Where separate files are used, these are placed at the head of each file. Red Hat already provides these for the services it installs with its distribution such as wu-fptd and swat. A default entry can be either on or off. For example, the chkconfig default and description entries for the ftp service are shown here:

```
# default: on
# description: The wu-ftpd FTP server serves FTP connections. It \
#    uses normal, unencrypted usernames and passwords for \
#    authentication.
```

Red Hat will indicate whether a service is set on or off by default. Should you want to turn on a service that is off by default, you will have to set its **disable** attribute to no, and restart **xinetd**. The Red Hat entry for the **wu-ftpd** FTP server is shown here. An initial comment tells us that it is on by default, but then the **disable** attribute turns it off.

```
# default: on
# description: The wu-ftpd FTP server serves FTP connections. It \
#    uses normal, unencrypted usernames and passwords for \
#    authentication.
service ftp
{
    socket_type     = stream
    wait            = no
    user            = root
    server          = /usr/sbin/in.ftpd
    server_args     = -l -a
    log_on_success  += DURATION USERID
    log_on_failure  += USERID
    nice            = 10
    disable         = yes
}
```

You can further add a variety of other attributes such as logging information about connections and server priority (nice). In the following example, the **log_on_success** attribute will log the duration (DURATION) and the user ID (USERID) for connections to a service, **log_on_failure** will log the users that failed to connect, and **nice** will set the priority of the service to 10.

```
log_on_success += DURATION USERID
log_on_failure += USERID
nice = 10
```

The default attributes defined in the defaults block often set global attributes such as default logging activity and security restrictions. **log_type** specifies where logging information is to be sent, such as to a specific file (FILE) or to the system logger (SYSLOG). log_on_success will specify information to be logged when connections are made, and **log_on_failure** will specify information to be logged when they fail.

```
log_type = SYSLOG authpriv
log_on_success = HOST PID
log_on_failure = HOST RECORD
```

For security restrictions, you can use **only_from** to restrict access by certain remote hosts. **no_access** denies access by the listed hosts, but no others. These controls take as their values IP addresses. You can list individual IP addresses, or a range of IP addresses, or a network using the network address. The instances attributes will limit the number of server processes that can be active at once for a particular service. The following examples restrict access to a local network 192.168.1.0 and the localhost, deny access from 192.168.1.15, and use the instances attribute to limit the number of server processes at one time to 60.

```
only_from = 192.168.1.0
only_from = localhost
no_access = 192.168.1.15
instances = 60
```

A sample default block is shown here:

```
defaults
{
 instances = 60
 log_type = FILE /var/log/servicelog
 log_on_success = HOST PID
 log_on_failure = HOST RECORD
 only_from = 192.168.1.0
 only_from = localhost
 no_access = 192.168.1.15
}
```

The **xinetd** program also provides several internal services including time, services, servers, and xadmin. The services service provides a list of currently active services, and servers provides information about servers. xadmin provides **xinetd** administrative support.

Instead of having one large **xinetd.conf** file, you can split it into several configuration files, one for each service. You do this by creating an **xinetd.conf** file with an **includedir** attribute that specifies a directory to hold the different service configuration files. In the following example, the **xinetd.d** directory will hold **xinetd** configuration files for services like **wu-ftpd**. Red Hat 7.0 uses just such an implementation. This approach has the advantage of letting you add services by just creating a new configuration file for it. Modifying a service only involves editing its configuration file, not an entire **xinetd.conf** file.

```
includedir /etc/xinetd.d
```

The following example shows the **xinetd.conf** file used for Red Hat Linux.

xinetd.conf

```
#
# Simple configuration file for xinetd
#
# Some defaults, and include /etc/xinetd.d/

defaults
{
    instances               = 60
    log_type                 = SYSLOG authpriv
    log_on_success      = HOST PID
    log_on_failure      = HOST
}

includedir /etc/xinetd.d
```

A few of the files in the **xinetd.d** directory are shown here. Notice that some are disabled by default, whereas others are not.

finger

```
# default: on
# description: The finger server answers finger requests. Finger is \
#  a protocol that allows remote users to see information such \
#  as login name and last login time for local users.
service finger
{
   socket_type    = stream
   wait           = no
   user           = nobody
   server         = /usr/sbin/in.fingerd
   disable        = yes
}
```

swat

```
# default: off
# description: SWAT is the Samba Web Admin Tool. Use swat \
#  to configure your Samba server. To use SWAT, \
#  connect to port 901 with your favorite web browser.
service swat
{
   port                = 901
   socket_type         = stream
   wait                = no
   only_from           = localhost
   user                = root
   server              = /usr/sbin/swat
   log_on_failure     += USERID
   disable             = yes
}
```

telnet

```
# default: on
# description: The telnet server serves telnet sessions; it uses \
# unencrypted username/password pairs for authentication.
service telnet
{
   flags               = REUSE
   socket_type         = stream
   wait                = no
   user                = root
   server              = /usr/sbin/in.telnetd
   log_on_failure     += USERID
   disable             = yes
}
```

wu-ftpd

```
# default: on
# description: The wu-ftpd FTP server serves FTP connections. It \
# uses normal, unencrypted usernames and passwords for authentication.
service ftp
{
   socket_type            = stream
   wait                   = no
   user                   = root
   server                 = /usr/sbin/in.ftpd
   server_args            = -l -a
   log_on_success        += DURATION USERID
   log_on_failure        += USERID
   nice                   = 10
   disable                = yes
}
```

TCP Wrappers

TCP wrappers add another level of security to **xinetd**-managed servers. In effect, the server is wrapped with an intervening level of security, monitoring connections and controlling access. A server connection made through **xinetd** is monitored verifying remote user identities and checking to make sure they are making valid requests. Connections are logged with the

syslogd daemon (Chapter 29) and may be found in **syslogd** files such as **/var/log/secure**. With TCP wrappers, you can also restrict access to your system by remote hosts. Lists of hosts are kept in the **hosts.allow** and **hosts.deny** files. Entries in these files have the format **service:hostname:domain**. The domain is optional. For the service, you can specify a particular service, such as FTP, or you can enter **ALL** for all services. For the hostname, you can specify a particular host or use a wildcard to match several hosts. For example, **ALL** will match on all hosts. Table 22-4 lists the available wildcards you can use. In the following example, the first entry allows access by all hosts to the Web service, **http**. The second entry allows access to all services by the **pango1.train.com** host. The third and fourth entries allow **rabbit.trek.com** and **sparrow.com** FTP access.

Table 22-3: xinetd Attributes

Attribute	**Description**
id	Identifies a service. By default, the service ID is the same as the service name.
type	Type of service: RPC, INTERNAL (provided by **xinetd**), UNLISTED (not listed in a standard system file).
flags	Possible flags include REUSE, INTERCEPT, NORETRY, IDONLY, NAMEINARGS (allows use of **tcpd**), NODELAY, DISABLE (disable the service). See the **xinetd.conf** Man page for more details.
disable	Specify "yes" to disable the service.
socket_type	Specify stream for a stream-based service, dgram for a datagram-based service, raw for a service that requires direct access to IP, and the seqpacket service for reliable sequential datagram transmission.
protocol	Specify a protocol for the service. The protocol must exist in **/etc/protocols**. If this attribute is not defined, the default protocol employed by the service will be used.
wait	Specifies whether the service is single-threaded or multithreaded (yes or no). If yes, the service is single-threaded, which means that **xinetd** will start the server and then stop handling requests for the service until the server stops. If no, the service is multithreaded and **xinetd** will continue to handle new requests for it.
user	Specifies the user ID (UID) for the server process. The user name must exist in **/etc/passwd**.
group	Specifies the GID for the server process. The group name must exist in **/etc/group**.
instances	Specifies the number of server processes that can be simultaneously active for a service.
nice	Specifies the server priority.
server	Specifies the program to execute for this service.
server_args	Lists the arguments passed to the server. This does not include the server name.
only_from	Controls the remote hosts to which the particular service is available. Its value is a list of IP addresses. With no value, service is denied to all.

Table 22-3: xinetd Attributes

Attribute	Description
no_access	Controls the remote hosts to which the particular service is unavailable.
access_times	Specifies the time intervals when the service is available. An interval has the form hour:min-hour:min.
log_type	Specifies where the output of the service log is sent, either syslog facility (SYSLOG) or a file (FILE).
log_on_success	Specifies the information that is logged when a server starts and stops. Information you can specify includes PID (server process ID), HOST (the remote host address), USERID (the remote user), EXIT (exit status and termination signal), and DURATION (duration of a service session).
log_on_failure	Specifies the information that is logged when a server cannot be started. Information you can specify includes HOST (the remote host address), USERID (user ID of the remote user), ATTEMPT (logs a failed attempt), RECORD (records information from the remote host to allow monitoring of attempts to access the server).
rpc_version	Specifies the RPC version for an RPC service.
rpc_number	Specifies the number for an UNLISTED RPC service.
env	Defines environment variables for a service.
passenv	The list of environment variables from **xinetd**'s environment that will be passed to the server.
port	Specifies the service port.
redirect	Allows a TCP service to be redirected to another host.
bind	Allows a service to be bound to a specific interface on the machine.
interface	Synonym for bind.
banner	The name of a file to be displayed for a remote host when a connection to that service is established.
banner_success	The name of a file to be displayed at the remote host when a connection to that service is granted.
banner_fail	Takes the name of a file to be displayed at the remote host when a connection to that service is denied.
groups	Allows access to groups the service has access to (yes or no).
enabled	Specifies the list of service names to enable.
include	Inserts the contents of a specified file as part of the configuration file.
includedir	Takes a directory name in the form of "includedir /etc/xinetd.d". Every file inside that directory will be read sequentially as an **xinetd** configuration file, combining to form the **xinetd** configuration.

```
http:ALL
ALL:pango1.train.com
ftp:rabbit.trek.com
ftp:sparrow.com
```

The **hosts.allow** file holds hosts to which you allow access. If you want to allow access to all but a few specific hosts, you can specify **ALL** for a service in the **hosts.allow** file, but list the ones you are denying access to in the **hosts.deny** file. Using IP addresses instead of hostnames is more secure because hostnames can be compromised through the DNS records by spoofing attacks where an attacker pretends to be another host.

Table 22-4: TCP Wrapper Wildcards

Wildcard	**Description**
ALL	Matches all hosts.
LOCAL	Matches any host specified with just a host name without a domain name. Used to match on hosts in the local domain.
UNKNOWN	Matches any user or host whose name or address is unknown.
KNOWN	Matches any user or host whose name or address is known.
PARANOID	Matches any host whose host name does not match its IP address.
EXCEPT	An operator that lets you provide exceptions to matches. It takes the form of *list1* **EXCEPT** *list2* where those hosts matched in *list1* that are also matched in *list2* are excluded.

When **xinetd** receives a request for an FTP service, a TCP wrapper monitors the connection and starts up the **in.ftpd** server program. By default, all requests are allowed. To allow all requests specifically for the FTP service, you would enter the following in your **/etc/hosts.allow** file. The entry **ALL:ALL** opens your system to all hosts for all services.

```
ftp:ALL
```

Note Originally, TCP Wrappers were managed by the **tcpd** daemon. However, **xinetd** has since integrated support for TCP Wrappers into its own program. You can explicitly invoke the **tcpd** daemon to handle services if you wish. The **tcpd** Man pages (**man tcpd**) provide more detailed information about **tcpd**.

Chapter 23: FTP Servers

Overview

The File Transfer Protocol (FTP) is designed to transfer large files across a network from one system to another. Like most Internet operations, FTP works on a client/server model. FTP client programs can enable users to transfer files to and from a remote system running an FTP server program. Chapter 19 discusses FTP clients. Any Linux system can operate as an FTP server. It only has to run the server software-an FTP daemon with the appropriate configuration. Transfers are made between user accounts on client and server systems. A user on the remote system has to log in to an account on a server and can then transfer files to and from that account's directories only. A special kind of user account, named *ftp,* allows any user to log in to it with the username "anonymous." This account has its own set of directories and files that are considered public, available to anyone on the network who wants to download them. The numerous FTP sites on the Internet are FTP servers supporting FTP user accounts with anonymous login. Any Linux system can be configured to support anonymous

FTP access, turning them into network FTP sites. Such sites can work on an intranet or on the Internet.

Note On Red Hat, the configuration files for anonymous FTP are in packages beginning with the term **anonftp**. Installing this package sets up your FTP directories and configures the FTP account.

FTP Daemons

FTP server software consists of an FTP daemon and configuration files. The *daemon* is a program that continuously checks for FTP requests from remote users. When a request is received, it manages a login, sets up the connection to the requested user account, and executes any FTP commands the remote user sends. For anonymous FTP access, the FTP daemon allows the remote user to log in to the FTP account using anonymous or ftp as the username. The user then has access to the directories and files set up for the FTP account. As a further security measure, however, the daemon changes the root directory for that session to be the FTP home directory. This hides the rest of the system from the remote user. Normally, any user on a system can move around to any directories open to him or her. A user logging in with anonymous FTP can only see the FTP home directory and its subdirectories. The remainder of the system is hidden from that user. This effect is achieved by the **chroot** operation (discussed later) that literally changes the system root directory for that user to that of the FTP directory. By default, the FTP server also requires a user be using a valid shell. It checks for a list of valid shells in the **/etc/shells** file. Most daemons have options for turning off this feature.

Several FTP server daemons are available for use on Linux systems. Most Linux distributions come with the Washington University FTP server called *wu-ftpd.* You can download RPM package updates for particular distributions from their FTP sites, such as **ftp.redhat.com**. The software package usually begins with the term **wu-ftpd**. You can obtain the original compressed archive from the Washington University archive at **http://wuarchive.wustl.edu/packages/wuarchive-ftpd**.

ProFTPD is a newer and popular FTP daemon based on an Apache Web server design. It features simplified configuration and support for virtual FTP hosts. Although it is not currently included with most distributions, you can download RPM packages from Red Hat. Check the **contrib** directories. The package begins with the term **proftpd**. The compressed archive of the most up-to-date version, along with documentation, is available at the ProFTPD Web site at **www.proftpd.net**. Another FTP daemon, ncftpd*,* is a commercial product produced by the same programmers who did the ncftp FTP client. ncftpd is free for academic use and features a reduced fee for small networks. Check **www.ncftpd.org** for more information.

Note Several security-based FTP servers are also available, including SSLftp and SSH sftp. SSLftp uses SSL (secure socket layer) to encrypt and authenticate transmissions, as well as MD5 digests to check the integrity of transmitted files. SSH sftp is an FTP server is now part of the Open SSH package, using SSH encryption and authentication to establish secure FTP connections.

Red Hat currently installs the wu-ftpd server and the **anon** anonymous FTP package during installation. At that time, an **ftp** directory along with several subdirectories are created where you can place files for FTP access. The directories have already been configured to control access by remote users, restricting use to only the **ftp** directories and any subdirectories. The **ftp** directory is placed in different directories by different distributions. On Red Hat, the **ftp** directory is placed in the **/var** directory, **/var/ftp**. Place the files you want to allow access to in the **ftp/pub** directory. For example, on Red Hat this would be at **/var/ftp/pub**. You can also create subdirectories and place files there. Once connected to a network, a remote user can connect to your system and download files you placed in **ftp/pub** or any of its subdirectories. The **anon** FTP package implements a default configuration for those directories and their files. You can change these if you want. If you are installing an FTP server yourself, you need to know the procedures detailed in the following sections to install an FTP server and create its data directories.

The **anon** FTP package does not create a directory where users can upload files to the FTP site. Such a directory is usually named the incoming directory located at **ftp/pub/incoming**. If you want such as directory, you will have to create it, make it part of the **ftp** group, and then set its permissions to allow users write access.

```
chgrp ftp /var/ftp/pub/incoming
chmod g+w /var/ftp/pub/incoming
```

Anonymous FTP: anon

An anonymous FTP site is essentially a special kind of user on your system with publicly accessible directories and files in its home directory. Anyone can log in to this account and access its files. Because anyone can log in to an anonymous FTP account, you must be careful to restrict a remote FTP user to only the files on that anonymous FTP directory. Normally, a user's files are interconnected to the entire file structure of your system. Normal users have write access that lets them create or delete files and directories. The anonymous FTP files and directories can be configured in such a way that the rest of the file system is hidden from them, and remote users are given only read access. In ProFTPD, this is achieved through configuration directives placed in its configuration file. An older approach used by wu-ftpd and implemented by the **anon** package involves having copies of certain system configuration, command, and libraries files placed within subdirectories of the FTP home directory. Restrictions placed on those sudirectories then control access by other users. Within the FTP home directory, you then have a publicly accessible directory that holds the files you want to make available to remote users. This directory usually has the name **pub**, for public.

An FTP site is made up of an FTP user account, an FTP home directory, and certain copies of system directories containing selected configuration and support files. Newer FTP daemons, such as ProFTPD, do not need the system directories and support files. Most distributions, including Red Hat, have already set up an FTP user account when you installed your system. On Red Hat, you can use the **anon** RPM package to set up the home directory and the copies of the system directories. If you do not have access to the **anon** package, you may have to create these system directories yourself.

The FTP User Account: Anonymous

To allow anonymous FTP access by other users to your system, you must have a user account named *FTP.* Red Hat has already created this account for you. If your system does not have such an account, you will have to create one. You can then place restrictions on the FTP account to keep any remote FTP users from accessing any other part of your system. You must also modify the entry for this account in your **/etc/passwd** file to prevent normal user access to it. The following is the entry you find in your **/etc/passwd** file on Red Hat systems that sets up an FTP login as an anonymous user:

```
ftp:*:14:50:FTP User:/var/ftp:
```

The asterisk in the password field blocks the account, which prevents any other users from gaining access to it, thereby gaining control over its files or access to other parts of your system. The user ID, 14, is a unique ID. The comment field is FTP User. The login directory is **/var/ftp**. When FTP users log in to your system, this is the directory in which they are placed. If a home directory has not been set up, create one and then change its ownership to the FTP user with the **chown** command.

The group ID is the ID of the **ftp** group, which is set up only for anonymous FTP users. You can set up restrictions on the **ftp** group, thereby restricting any anonymous FTP users. Here is the entry for the **ftp** group you find in the **/etc/group** file. If your system does not have one, you should add it.

```
ftp::50:
```

Anonymous FTP Server Directories

As previously noted, on Red Hat the FTP home directory is named **ftp** and is placed in the **/var** directory. When users log in anonymously, they are placed in this directory. An important part of protecting your system is preventing remote users from using any commands or programs not in the restricted directories. For example, you would not let a user use your **ls** command to list filenames because **ls** is located in your **/bin** directory. At the same time, you want to let the FTP user list filenames using an **ls** command. Newer FTP daemons like ProFTPD solve this problem by creating secure access to needed system commands and files, while restricting remote users to only the FTP site's directories. Another more traditional solution, used by wu-ftpd, is to create copies of certain system directories and files needed by remote users and to place them in the **ftp** directory where users can access them. A **bin** directory is placed in the **ftp** directory and remote users are restricted to it, instead of the system's **bin** directory. Whenever they use the **ls** command, remote users are using the one in **ftp/bin**, not the one you use in **/bin**.

On Red Hat, the **anon** RPM package will set up these copies of system directories and files. Otherwise, you may have to create these directories and support files yourself. On Red Hat, the **anon** package installs **etc**, **bin**, and **lib** directories in the **/var/ftp** directory. These contain localized versions of system files needed to let an FTP client execute certain FTP commands, such as listing files or changing directories. The **ftp/etc** directory contains versions of the password and group configuration files, the **ftp/bin** directory contains copies of shell and compression commands, and the **ftp/lib** directory holds copies of system libraries. The directories set up by the **anon** package are shown here:

```
ftp
ftp/bin
ftp/etc
ftp/lib
ftp/pub
```

The **ftp/etc** directory holds a version of your **passwd** and **group** files specially configured for FTP access. Again, the idea is to prevent any access to the original files in the **/etc** directory by FTP users. The **ftp/etc/passwd** file should not include any entries for regular users on your system. All entries should have their passwords set to * to block access. The **group** file should not include any user groups, and all passwords should be set to *.

ftp/etc/passwd

```
root:*:0:0:::
bin:*:1:1:::
operator:*:11:0:::
ftp:*:14:50:::
nobody:*:99:99:::
```

ftp/etc/group

```
root::0:
bin::1:
daemon::2:
sys::3:
adm::4:
ftp::50:
```

If, for some reason, you do not have access to the **anon** package, you can set up the anonymous FTP directories yourself. Again, remember, if you are using ProFTPD, you do not need any of these files, except for an FTP home directory. You must use the **chmod** command to change the access permissions for the directories so remote users cannot access the rest of your system. Create an **ftp** directory and use the **chmod** command with the permission 555 to turn off write access: **chmod 555 ftp**. Next, make a new **bin** directory in the **ftp** directory, and then make a copy of the **ls** command and place it in **ftp/bin**. Do this for any commands you want to make available to FTP users. Then create an **ftp/etc** directory to hold a copy of your **passwd** and **group** files. Again, the idea is to prevent any access to the original files in the **/etc** directory by FTP users. The **ftp/etc/passwd** file should be edited to remove any entries for regular users on your system. All other entries should have their passwords set to * to block access. For the **group** file, remove all user groups and set all passwords to *. Create an **ftp/lib** directory, and then make copies of the libraries you need to run the commands you placed in the **bin** directory.

Anonymous FTP Files

A directory named **pub**, located in the FTP home directory, usually holds the files you are making available for downloading by remote FTP users. When FTP users log in, they are placed in the FTP home directory (**/var/ftp** on Red Hat), and they can then change to the **pub** directory to start accessing those files (**/var/ftp/pub** on Red Hat). Within the **pub** directory, you can add as many files and directories as you want. You can even designate some directories as upload directories, enabling FTP users to transfer files to your system.

Note In each subdirectory set up under the **pub** directory to hold FTP files, you should create a **Readme** file and an **index** file as a courtesy to FTP users. The **Readme** file contains a brief description of the kind of files held in this directory. The **index** file contains a listing of the files and a description of what each one holds.

Permissions

Technically, any remote FTP user gaining access to your system is considered a user and, unless restricted, could access other parts of your file system, create directories and files, or delete the ones already there. Permissions can be used to restrict remote users to simple read access, and the rest of your file system can be hidden from the FTP directories. The **anon** package and the ProFTPD daemon already implement these restrictions. If you are manually creating your anonymous FTP files, you must be sure to set the permission correctly to restrict access.

Normally, a Linux file structure interconnects all the directories and files on its system. Except where prevented by permissions set on a directory or file, any user can access any directory or file on your system. Technically, any remote FTP user gaining anonymous access is an anonymous user and, as a user, could theoretically access an unrestricted directory or file on your system. To restrict FTP users to the FTP home directory, such as **ftp**, and its subdirectories, the rest of the file structure must be hidden from them. In effect, the FTP home directory should appear to be the root directory as far as FTP users are concerned. On Red Hat, the FTP home directory **ftp** would appear to the remote FTP user as the root directory. The real root directory, /, and the rest of the directory structure remain hidden. The FTP daemon attains this effect by using the **chroot** command to make the FTP home directory appear as a root directory, with the FTP user as the argument. When a remote FTP user issues a **cd** / command to change to the root, they always change to the FTP home directory, not the system's root directory. For example, on Red Hat, the **cd** / command would change to **ftp**.

As a further restriction, all the directories that hold commands in the FTP home directory, as well as the commands themselves, should be owned by the root, not by the FTP user. In other words, no FTP user should have any control over these directories. The root has to own the FTP home directory's **bin** and **etc** subdirectories and all the files they contain (**/var/ftp/bin** and **/var/ftp/etc** on Red Hat). The **anon** package already has set the ownership of these directories to the root. If you need to set them manually, you can use the **chown** command. The following example changes the ownership of the **/var/ftp/bin** directory to the root:

```
# chown root  /var/ftp/bin
```

Permissions for the FTP directories should be set to allow access for FTP users. You recall that three sets of permissions exist-read, write, and execute for the owner, the group, and others. To allow access by FTP users, the group and other permissions for directories should be set to both read and execute. The execute permission allows FTP users to access that directory, and the read permission allows listing the contents of the directory. Directories should not allow write permission by FTP users. You don't want them to be able to delete your directories or make new ones. For example, the FTP **bin** directory needs both read and execute permissions because FTP users have to access and execute its commands. This is particularly true for directories, such as **pub**, which hold the files for downloading. It must have both read and execute permissions set.

You, as the owner of the directories, may need write permission to add new files or subdirectories. Of course, you only need this when you are making changes. To add further security, you could set these directories at just read and execute, even for the owner when you are not making changes. You can set all permissions to read and execute with the **chmod** command and the number 555 followed by the directory name. This sets the owner, group, and other permissions to read and execute. The permissions currently in place for the FTP directories set up by the **anon** package are designated by the number 755, giving the owner write permission.

```
# chmod 555 /var/ftp/bin
```

Permissions for files within the FTP **bin** directory and other special FTP directories can be more restrictive. Some files only need to be read, while others must be execute. Files in the FTP **bin** or **lib** directories only have to be execute. These could have their permissions set to 555. Files in the FTP **etc** directory such as **passwd** and **group** should have their permissions set to 111. They only have to be read. You always use the **chmod** command to set permissions for files, as shown in the following example. The **adnon** package sets these permissions at read and execute, 555.

```
# chmod 111 /var/ftp/etc/passwd
```

FTP Server Tools

Both the wu-ftpd and ProFTPD daemons provide a set of FTP tools you can use to manage your FTP server. With the **ftpshut** command, you can smoothly shut down a running server, warning users of the shutdown well before it happens. **ftpwho** can tell you who is currently connected and what they are doing. **ftpcount** can give you the number of connections currently in effect. Although each daemon has its own set of tools, they perform the same action with much the same set of options. Tools provided by both ProFTPD and wu-ftpd have the same name and options, though ProFTPD provides more information on virtual hosts and has some added options.

ftpshut

With the **ftpshut** command, you can have the FTP server shut down at a given time, rather than suddenly shutting it down by killing its process. This gives you the chance to warn users the server is shutting down and not to start any long downloads. **ftpshut** takes several options for specifying the time and including a warning message. **ftpshut** takes as its arguments the time until the shutdown, followed by the warning message you want sent to users. The time can be a word such as "now" that effects an immediate shutdown, a + sign with the number of minutes remaining, or a specific time of day indicated by an HHMM format, where HH is the hour in a 24-hour cycle and MM is the minute. The following example shuts down the FTP server in ten minutes, issuing a warning to users:

```
ftpshut +10  "Shutdown in ten minutes"
```

Shutdown disables new FTP access ten minutes before a scheduled shutdown, though this can be changed using the **-l** option with the number of minutes you want. Five minutes before a scheduled shutdown, all current connections are disconnected. You can adjust the time with the **-d** options. The warning message is formatted at 75 characters, and you can use special formatting symbols for in-place substitutions of certain values in the warning message, such as the shutdown time. These symbols are called *magic cookies.* For example, **%s** is the shutdown time, **%r** is the time when new connections are refused, **%d** is the time when current connections are cut, **%M** is the maximum number of users, and **%L** is the local hostname.

ftpwho and ftpcount

With the **ftpwho** command, you can find out who is currently connected to your FTP server. **ftpwho** shows the current process information for each user. The output displays five fields: the process ID, the tty connection, the status of the connection, the amount of CPU time used so far for the process, and the connection details. The status of the connection is *R* for running, *S* for sleeping, and *Z* for crashed. The connection details include the Internet address from where the connection is made, the user making the connection, and the task currently being performed, such as downloading a file. The field begins with the name of the FTP daemon, usually **ftpd**, followed by the different segments separated by colons.

ftpcount displays the number of users connected to your FTP server, broken down according to the classes specified in your **.ftpaccess** file. Along with the number of users, it shows the maximum number allowed to connect.

The Washington University FTP daemon: wu-ftpd

The Washington University FTP daemon is currently the most widely used FTP server on Linux systems. It is the FTP server installed by most Linux distributions. The name of the Washington University FTP daemon is wu-ftpd. The wu-ftpd options are shown in Table 23-1. wu-ftpd must be running to allow FTP access by remote users. As with other servers, you can start the FTP server at boot time, through **xinetd** when a request is received, or directly from the command line. By default, the wu-ftpd server is installed to run using **xinetd**. The use of **xinetd** for the servers is described in detail in the previous chapter. The command name for the FTP server invoked by **xinetd** is **in.ftpd.** This is a link to the **wu-ftpd** command. **xinetd** will run a file called **wu-ftpd** located in the **/etc/xinetd.d** directory. A copy of the script is shown here.

wu-ftpd

```
# default: on
# description: The wu-ftpd FTP server serves FTP connections. It uses \
# normal, unencrypted usernames and passwords for authentication.
service ftp
{
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /usr/sbin/in.ftpd
        server_args             = -l -a
        log_on_success         += DURATION USERID
        log_on_failure         += USERID
        nice                    = 10
        disable                 = yes
}
```

Initially, the server will be turned off. You can turn it on with the **chkconfig** command and the **on** argument, as shown here. Use the **off** argument to disable the server.

```
chkconfig wu-ftpd on
```

Restart **xinetd** with the **service** command to restart the wu-ftpd server, should you make configuration changes.

```
service xinetd restart
```

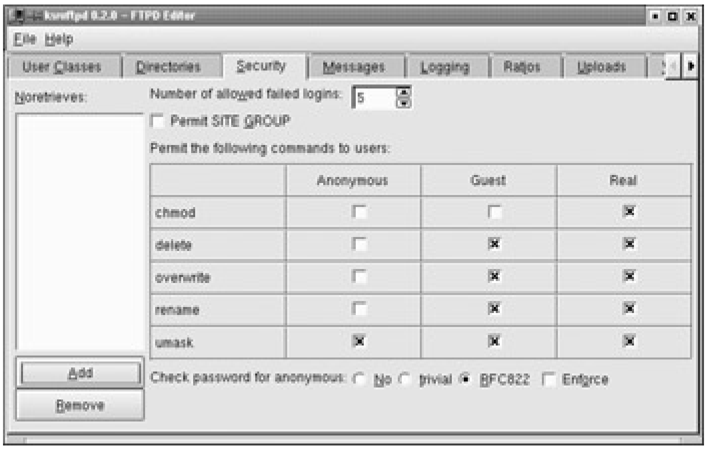
If you want to run your server continually (like a Web server), you have to configure it to start up initially with your system. On Red Hat, this means creating an **init** script for it in the **/etc/rc.d/init.d** directory, so it starts when you boot your system. You can also start the FTP server directly from the command line by entering the **wu-ftpd** command with any options or arguments. The wu-ftpd server can be called with several options. Usually, it is called with the **-l** option that allows logins. The **-t** and **-T** options set timeouts for users, cutting off those that have no activity after a certain period of time. The **-d** option displays debugging information, and **-u** sets the umask value for uploaded files.

Table 23-1: wu-ftpd Options

Option	Effect
-d	Writes debugging information to the **syslog**
-l	Logs each FTP session in the **syslog**
-t*seconds*	Sets the inactivity timeout period to specified seconds (default is 15 minutes)
-T*seconds*	The maximum timeout period allowed when timeout is set by user (default is two hours)
-a	Enables use of the **ftpaccess** configuration file
-A	Disables use of the **ftpaccess** configuration file
-L	Logs commands sent to the **ftpd** server to the **syslog**
-I	Logs files received by **ftpd** to **xferlog**
-o	Logs files transmitted by **ftpd** to the **syslog**

Configuring the wu-ftpd Server with kwuftpd

Red Hat 7.1 includes a GUI wu-ftpd configuration tool for the KDE desktop called kwuftpd. Though still under development, current versions can be used to configure your **ftpaccess** file easily. On Red Hat, you can access kwuftpd from the KDE desktop. Kwuftpd presents a set of tabbed panels for tasks like controlling users and uploads, specifying server directories, and setting up virtual hosts. Figure 23-1 shows the Security panel where you can specify files that can't be accessed, control access to basic commands by real, guest, and anonymous users, and specify the allowable number of failed logins.



Number of allowed failed logins: 5

☐ Permit SITE GROUP

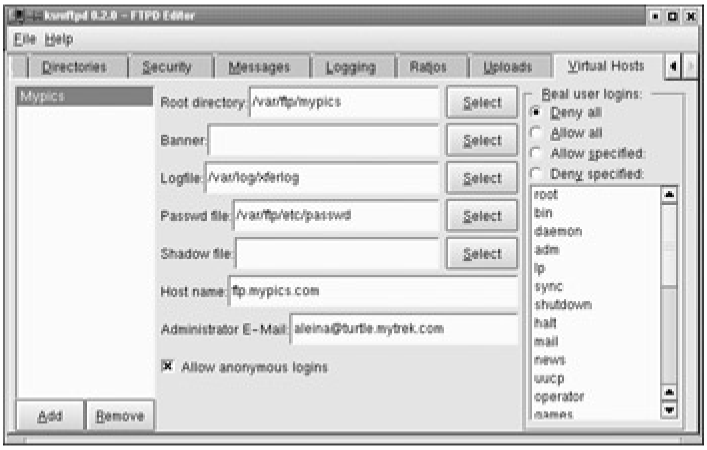
Permit the following commands to users:

	Anonymous	Guest	Real
chmod	☐	☐	☒
delete	☐	☒	☒
overwrite	☐	☒	☒
rename	☐	☒	☒
umask	☒	☒	☒

Check password for anonymous: ○ No ○ trivial ● RFC822 ☐ Enforce

Figure 23-1: kwuftpd

With kwuftpd, you can also set up virtual hosts. On the Virtual Hosts panel, shown in Figure 23-2, you can specify the virtual host information such as the hostname, root directory, and log file. Click on the Add button to create a Virtual Host entry, then enter its information. You can also control access by real users.



Root directory: /var/ftp/mypics

Banner:

Logfile: /var/log/xferlog

Passwd file: /var/ftp/etc/passwd

Shadow file:

Host name: ftp.mypics.com

Administrator E-Mail: aleina@turtle.mytrek.com

☒ Allow anonymous logins

Real user logins:
● Deny all
○ Allow all
○ Allow specified:
○ Deny specified:

root
bin
daemon
adm
lp
sync
shutdown
halt
mail
news
uucp
operator
games

Figure 23-2: Virtual hosts with kwuftpd

Note You can also use Linuxconf and Webmin to configure the wu-ftpd server. Linuxconf requires that you load its wu-ftpd module.

wu-ftpd Server Configuration Files

You can use numerous configuration options to tailor your FTP server to your site's particular needs. wu-ftpd makes use of several configuration files located in the system's **/etc** directory. All begin with the pattern **ftp**. The primary configuration file is named **ftpaccess**. Here, you provide basic server information and access for specified directories. The **ftphosts**, **ftpusers**, and **ftpgroups** control access by systems, particular users, and groups. **ftpconversions** specifies how archive and compression operations are to be performed on files before or after they are transferred. **xferlog** is the log file that stores a running log of all transactions performed by the server.

ftpaccess

The **ftpaccess** file determines capabilities users have after they gain access to your FTP site. Access, information, permissions, logging, and several miscellaneous capabilities can be designated. You can have entries that create aliases for certain directories, display a message when FTP users log in, or prevent anonymous users from deleting files. A **loginfails** entry determines the number of login tries a user can make before being cut off, and the **email** entry specifies the e-mail address of the FTP administrator. The Man page for **ftpaccess** lists the possible entries. The **ftpaccess** file with the configuration used on Red Hat systems is shown in this section. For commonly used **ftpaccess** entries, see Table 23-2. For more detailed information, check the **ftpaccess** Man page and wu-ftpd documentation.

In the **ftpaccess** file, you set capabilities for different types of users, called *classes*. Three different types of users exist: anonymous, guest, and real. *Anonymous users* are any users using the anonymous login name. *Guest users* are those given special guest access with **guestgroup** or **guestuser** options. A *real user* is one who has an account on the system and is using an FTP connection to access it. You can define your own class using the **class** option. In the **ftpaccess** file shown here, a class called **all** is created that consists of all users of the anonymous, guest, and real types.

The message entry specifies a file with the message to be displayed and when that message is to appear. You can have one message appear when users log in and other messages displayed when users enter certain directories. For example, the following entry will display the message in the **/welcome.msg** file when a user logs in:

```
message /welcome.msg            login
```

Table 23-2: /etc/ftpaccess wu-ftpd Configuration File

Access Capabilities	**Description**
autogroup *group classglob* [*classglob...*]	This allows access to a group's read-only files and directories by particular classes of anonymous users. *group* is a valid group from **/etc/group**.
class *class* typelist addrglob [*addrglob...*]	Defines *class* of users, with source addresses of the form *addrglob. typelist* is a comma-separated list of the user types: anonymous, guest, and real.
deny *host-addrglob message_file*	Always deny access to host(s) matching *host-addrglob. message_file* is displayed.
guestgroup *groupname* [*groupname...*]	Allow guest access by a real user, where the user is a member of the specified group. A password entry for the guest user specifies a home directory within the FTP site directories.
guestuser *usrname* [*userpname...*]	Allow guest access by a real user.
limit *class* n times message_file	Limit class to *n* users at times-*times*, displaying *message_file* if access is denied.
noretrieve *file-list*	Deny retrieval ability of these files.
loginfails *number*	After *number* login failures, terminate the FTP connection. Default value is 5.
Private *yes|no*	The user becomes a member of the group specified in the group access file **ftpgroups**.
Informational Capabilities	
banner *file*	The banner is displayed before login. File requires full

Table 23-2: /etc/ftpaccess wu-ftpd Configuration File

Access Capabilities	**Description**
	pathname.
email *email-address*	Defines the e-mail address of the FTP manager.
message *file* { *when* { *class* ...}}	FTP displays the contents of the *file* at login time or upon changing directories. The **when** parameter may be LOGIN or CWD=*dir; dir* specifies the directory that displays the message when entered. Magic cookies can be in the message file that cause the FTP server to replace the cookie with a specified text string, such as the date or the username.
readme *file* { *when* { *class*}}	The user is notified at login time or upon using a change working directory command (**cd**) that *file* exists and was modified on such and such date.
Logging Capabilities	**Description**
log commands *typelist*	Enables logging of individual commands by users.
log transfers *typelist directions*	Enables logging of file transfers. *directions* is a comma-separated list of the terms "inbound" and "outbound," and logs transfers for files sent to the server and sent from the server.
Miscellaneous Capabilities	
alias *string* dir	Defines an alias, *string*, for a directory.
cdpath *dir*	Defines an entry in **cdpath**. This defines a search path used when changing directories.
compress yes|no *classglob* [*classglob* **tar** yes|no *classglob* [*classglob*...]...]	Enables **compress** or **tar** capabilities for any class matching of *classglob*. The actual conversions are defined in the external file **ftconversion**.
shutdown *path*	If the file pointed to by *path* exists, the server checks the file regularly to see if the server is going to be shut down.
virtual *address* root|banner|logfile *path*	Enables the virtual FTP server capabilities. Specify the root, banner, and logfile files.
virtual *address* hostname|email *string*	Specify the hostname and e-mail for the virtual host.
virtual *address* allow *user-list*	List users allowed access to the virtual host.
virtual *address* deny *user-list*	List users denied access to the virtual host.
virtual *address* private	Deny access by anonymous users
virtual *address* password *path*	Use a different password file for the virtual host.
virtual *address* shadow *path*	Use a different shadow password file for the virtual host.
Permission Capabilities	Allows or disallows the ability to perform the specified function. By default, all users are allowed.
chmod yes | no *typelist*	Allow or disallow changing file permissions.

Table 23-2: /etc/ftpaccess wu-ftpd Configuration File

Access Capabilities	**Description**
delete yes | no *typelist*	Allow or disallow deleting files, **rm**.
Permission Capabilities	**Description**
overwrite yes | no *typelist*	Allow or disallow modifying files.
rename yes | no *typelist*	Allow or disallow renaming files, **mv**.
umask yes | no *typelist*	Allow or disallow file creation permissions.
passwd-check *none* | *trivial* | *rfc822* (*enforce* | *warn*)	Define the level and enforcement of password checking done by the server for anonymous FTP. *rfc822* requires an e-mail address in a valid e-mail address format.
path-filter *typelist* mesg allowed_charset { *disallowed* regexp...}	For users in *typelist*, **path-filter** defines regular expressions that control what a filename can or cannot be. Multiple disallowed *regexps* may occur.
upload *root-dir* dirglob yes|no *owner* group *mode* ["dirs"|"nodirs"]	Define a directory with *dirglob,* which permits or denies uploads.

To set permissions, you use the command followed by a yes or a no and then a list of the user types or classes. An example of the **ftpaccess** file on Red Hat systems is shown here. In this example, all users can perform **tar** and **compress** operations, but anonymous and guest users are prohibited from using **chmod**, **delete**, **overwrite**, and **rename** operations. They also cannot erase files, modify them, or change their names or permissions.

/etc/ftpaccess

```
# define the class of users
class   all   real,guest,anonymous  *
# Email address of FTP managerm
email root@localhost
# Allow only 5 login failures per connection
loginfails 5
# Login and change directory README files
readme  README*    login
readme  README*    cwd=*
# Login and change directory message files
message /welcome.msg     login
message .message                cwd=*

# Set permissions
# Allow access to compress and tar operations.
compress  yes        all
tar              yes        all
# Deny access by guests and anonymous users to commands
chmod       no          guest,anonymous
delete      no          guest,anonymous
overwrite   no          guest,anonymous
rename      no          guest,anonymous

# Log file transfers, shutdown notice, password email address
log transfers anonymous,real inbound,outbound
```

```
shutdown /etc/shutmsg
passwd-check rfc822 warn
```

You can further modify your **ftpaccess** file to control access to particular users or groups of users by defining a class for them and placing restrictions on that class. For example, you could define a class for connections from a particular host as shown here:

```
class  manycons   192.168.1.55  *
```

Then apply controls such as limiting the number of connections from that host to five. This would be helpful for cases where you know many users from a particular host are trying to use your FTP site.

```
limit manycons  5  Any
```

If you want to allow users to upload files to a specific directory, you will need to specify that directory with an **upload** entry. For the **upload** entry, you need to specify the root directory of the FTP site (root directory for the **ftp** user), the directory to which files can be uploaded, whether uploads are permitted (**yes|no**), and the owner, group, and the file permissions for the files in that directory. The **dirs** and **nodirs** options will allow and disallow the user from creating subdirectories in that upload directory. In the next example, the **/var/ftp/pub/incoming** directory is the directory to which users can upload copies (**/var/ftp** is the FTP root directory). Any files in that directory will be owned by the **ftp** user, which is part of the **ftp** group, and will have the permissions 0666 (read/write access). Users are also not allowed to create subdirectories.

```
upload  /var/ftp   /pub/incoming   yes  ftp  ftp    066   nodirs
```

The wu-ftpd server also supports virtual hosts. These are FTP sites that have different addresses but use the same server. To enable wu-ftpd to service a virtual host, you need to specify information about its root directory, its hostname, and user access. You specify this information with the **virtual** command using different options. The first argument is usually the IP address of the virtual FTP server. Then, different options let you specify certain information. With the **root** option you can specify the server's root directory. The **banner**, **password**, and **logfile** options specify the banner, password, and log files. With the **hostname** and **email** options, you give the virtual host's hostname and the e-mail address for its administrator. The **deny** and **allow** options let you list read and guest users that can or cannot have access to the virtual server. The **private** option denies access to anonymous users.

In the following example, a virtual FTP server called **ftp.mypics.com** is created whose IP address is at 10.0.0.1. This virtual server has a root directory at **/var/ftp/mypics** and uses the log file at **/var/log/syslog**. The administrator's e-mail address is aleina@turtle.mytrek.com.

```
virtual 10.0.0.1 hostname ftp.mypics.com
virtual 10.0.0.1 root  /var/ftp/mypics
virtual 10.0.0.1 logfile  /var/ftp/syslog
virtual 10.0.0.1 email  aleina@turtle.mytrek.com
```

ftphosts

You use the **ftphosts** file to allow or deny access by other host computers to your FTP site. When the remote system accesses your system, it does so by logging in as a registered user. Access is made through a user account already set up on your system. You allow the remote host to log in as a certain specific user or deny access as a certain user. You could use this kind of control to allow or deny anonymous access to the FTP user by a remote host.

The file **ftphosts** has two kinds of entries: one for allowing access and the other for denying access. Entries to allow access begin with the keyword **allow**, then the user account on your system to which the host is allowed access, followed by the address of the remote host. The address can be a pattern that can be used to match several hosts. You can use any of the filename generation symbols (see Chapter 9). Entries to deny access begin with the keyword **deny**, then the user account on your system to which the host is denied access, followed by the address of the remote host. The terms **deny** and **allow** can be misleading. **allow** is a much more restrictive control, whereas **deny** is a much more open control. **allow** only allows access from the remote host to the specified account. No other access is permitted. You could use **allow** to permit a remote host anonymous access only. **deny**, on the other hand, only denies access to the specified account. You could use **deny** to deny anonymous access by a certain system, but not any direct FTP access from one user to another.

ftpusers and ftpgroups

The **ftpusers** files list users that cannot access the FTP service. For example, the root user should not be accessible through an FTP connection, even if you knew the password. This file will initially hold a listing of all your system users such as **root**, **mail**, and **bin**. **ftpgroups** is a group access file that allows FTP users to become members of specified groups on your system. This file lists special group passwords. For these to work, the **Private** entry must be set to **yes** in the **ftpaccess** file.

ftpconversions

The **ftpconversions** file holds possible FTP conversions for compression and archive operations. It operates as an FTP conversions database, listing all possible conversions. A default **ftpconversions** file is included with the installation package that already has entries for the most common conversion operations. Each line in the file is a record of eight fields, with the fields separated by colons. The fields are Strip Prefix and Postfix, Addon Prefix and Postfix, External Command, Types, Options, and Description. The Prefix and Postfix fields refer to changes made to the filename after the specified action is performed. The Strip Postfix removes a specified suffix from a filename, and the Add Postfix adds a suffix. For example, a gzipped compressed file has a suffix of **.gz**. If the command is to compress a file with gzip, then the Add Postfix entry should have the **.gz** placed in it. When the file is compressed, **.gz** is added to the end. If you were decompressing a file with gunzip, then you would want to remove the **.gz** suffix. For this, you would place **.gz** in the Strip Postfix field. The Strip and Add Prefix fields perform the same kind of action, but for prefixes.

The **external** command is the command you would use to convert the file. You can list command options after the command. The filename you are operating on is specified with **%s**, usually placed after any options. For example, you would use the **tar** command to extract a **.tar** archived file and **gunzip** to decompress a **.gz** file. The Type field lists the type of files that can be operated on by the command. These can be regular files, character files, or directories, as indicated by the entries **T_REG**, **T_ASCII**, and **T_DIR**. You can specify more than one entry by placing a **|** between them. The Options field specifies the type of operation the command performs. Currently, options exist for compression, decompression, and use of the **tar** command: **O_COMPRESS, O_UNCOMPRESS,** and **O_TAR**. You can list more than one by separating them with a **|** symbol. The description provides some documentation as to what the conversion operation does. Here is the **ftpconversion** file used on Red Hat systems:

```
:.Z:  :  :/bin/compress -d -c %s:T_REG|T_ASCII:O_UNCOMPRESS:UNCOMPRESS
:   : :.Z:/bin/compress -c %s:T_REG:O_COMPRESS:COMPRESS
:.gz: :  :/bin/gzip -cd %s:T_REG|T_ASCII:O_UNCOMPRESS:GUNZIP
:   : :.gz:/bin/gzip -9 -c %s:T_REG:O_COMPRESS:GZIP
:   : :.tar:/bin/tar -c -f - %s:T_REG|T_DIR:O_TAR:TAR
:   : :.tar.Z:/bin/tar -c -Z -f -
                             %s:T_REG|T_DIR:O_COMPRESS|O_TAR:TAR+COMPRESS
:   : :.tar.gz:/bin/tar -c -z -f -
                             %s:T_REG|T_DIR:O_COMPRESS|O_TAR:TAR+GZIP
```

FTP log file: xferlog

This file contains log information about connections and tasks performed by your FTP server. On Red Hat systems, this file is found in the **/var/log** directory. On other systems, this file may be on the **/usr/adm** directory. The file is made up of server entries, one on each line. The entry is divided into several fields separated by spaces. The fields are current-time, transfer-time, remote-host, file-size, filename, transfer-type, special- action-flag, direction, access-mode, username, service-name, authentication-method, and authenticated-user-id. The *transfer-time* is the time in seconds for the transfer. The *remote-host* is the address of the remote system making the connection, and *username* is the name of the user on that system. The *transfer-type* is either an *a* for ASCII or *b* for binary. The *access-mode* is the method by which the user logged in: *a* for anonymous, *g* for guest, and *r* for a real login (to another account on your system). The direction is either *o* for outgoing or *i* for incoming. The following example shows the file **mydoc** transferred by user **larisa**. The entry first shows the time, the FTP server host, and the size of the file along with the filename. The file transfer was binary, outgoing, and made by an anonymous user (b o a). The user's name (password) and the service used, **ftp**, are also shown.

```
Sun Oct 29 11:15:40 2000 1 turtle.mytrek.com 55945
/var/ftp/pub/mydoc b o a larisa@turtle.mytrek.com ftp 0 * c
```

Professional FTP Daemon: ProFTPD

ProFTPD is based on the same design as the Apache Web server, implementing a similar simplified configuration structure and supporting such flexible features as virtual hosting. ProFTPD RPM packages are available from Red Hat and from **www.proftpd.net**. Unlike other FTP daemons, you do not need to set up special subdirectories of system files in the FTP home directory. No special **bin** or **etc** files are needed. You can also set up ProFTPD to alternate automatically between xinetd startups or as a standalone server constantly running, depending on the system load.

ProFTPD's tools operate in the same way as the wu-ftpd tools. **ftpshut** shuts down the system at specified times with warnings. With ProFTPD, you can shut down a virtual host while the main server continues to run. **ftpwho** displays a list of all remote users currently connected, broken down according to virtual hosts and servers. **ftpcount** shows the number of current connections by server and virtual hosts. See the previous section on FTP tools for more information.

install and startup

If you install ProFTPD using distribution RPM packages such as the one for Red Hat, the required configuration entries are made in your **proftpd.conf** files. If you installed from compiled source code, you may have to modify the entries in the default **proftpd.conf** file provided. Make sure the FTP user and group specified in the **proftpd.conf** file actually exist.

You can download ProFTPD from the ProFTPD Web site at **www.proftpd.net**. The RPM package version will contain three packages: a core application, a standalone version, and an inetd version that will run on xinetd. Download and install the core package first. Then, depending on whether you want to run ProFTPD as a standalone or inetd process, install either the inetd or the standalone package.

```
proftpd-core-1.2.0pre10-1.i686.rpm
proftpd-inetd-1.2.0pre10-1.i686.rpm
proftpd-standalone-1.2.0pre10-1.i686.rpm
```

You can run ProFTPD either as a standalone process or from xinetd. Make sure the appropriate entry is made in the ServerType directive in your **proftpd.conf** file. The standalone RPM package will install **proftpd** to run as a standalone server, setting the ServerType to standalone. **proftpd** options are listed in Table 23-3. On Red Hat systems, a startup script named **proftpd** is placed in the **/etc/rc.d/inet.d** directory that starts up the daemon when you boot your system. A standalone process is continually running. You can start, stop, and restart the server using the **service** command, as shown here:

```
service proftpd restart
```

To set the runlevels at which it will start automatically, you can use **chkconfig**, Sys V Runlevel Editor, or Sys V Init Editor tools. The following command sets **proftpd** to run automatically from runlevels 3 and 5:

```
chkconfig --level 35 proftpd on
```

Table 23-3: ProFTPD Daemon Startup Options

Option	**Description**
-h,--help	Use description, including options.
-n,--nodaemon	Runs the **proftpd** process in standalone mode (must also specify standalone as ServerType in the configuration file).
-v,--version	Display ProFTPD version number.
-d, --debug *debuglevel*	Sets **proftpd**'s internal debug level (1-5).
-c,--config *config-file*	Specifies alternate configuration file.
-p,--persistent 0|1	Disables (0) or enables (1) the default persistent password support, which is determined at configure time for each platform (see PersistentPasswd directive).

Table 23-3: ProFTPD Daemon Startup Options

Option	Description
-l,--list	Lists all modules compiled into **proftpd**.

The following command would disable **proftpd**:

```
chkconfig --del proftpd
```

If you want to run **proftpd** as an **xinetd** process, you first must change the ServerType to inetd and disable the **proftpd** startup script in the **/etc/rc.d/inet.d** directory. To run ProFTPD from **xinetd**, make sure to create an appropriate file for it in the **/etc/xinetd.d directory**. Currently, the **proftpd** inetd RPM package implements an **in.proftpd** link to the **proftpd** daemon. Use this link to invoke ProFTPD in the ProFTPD **xinetd** file. A simple ProFTPD **xinetd** file, named **proftpd**, would look like this:

```
service proftp
{
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /usr/sbin/in.proftpd
        disable                 = no
}
```

Use the **chkconfig** command with the **off** and **on** arguments to disable or enable **proftpd** running under **xinetd**.

```
chkconfig proftpd on
```

proftpd.config and .ftpaccess

ProFTPD uses only one configuration file, named **proftpd.conf**, located in the **/etc** directory. Configuration entries take the form of directives. This format is purposely modeled on Apache configuration directives. With the directives, you can enter basic configuration information, such as your server name, or perform more complex operations, such as implementing virtual FTP hosts. The design is flexible enough to enable you to define configuration features for particular directories, users, or groups.

To configure a particular directory, you can use an **.ftpaccess** file with configuration options placed within that directory. These **.ftpaccess** options take precedence over those in the **proftpd.conf** directory. **.fptaccess** files are designed to operate like **.htaccess** files in the Apache Web server that configure particular Web site directories. You can find a complete listing of ProFTPD configuration parameters at the ProFTPD Web site (**www.proftpd.net**) and in the ProFTPD documentation installed in **/usr/doc** as part of the ProFTPD software package. Several of the more commonly used parameters are listed in Table 23-4. When creating a new configuration, you should make a copy of the **proftpd.conf** configuration file and modify it. Then you can test its syntax using the **proftpd** command with the **-c** option and the name of the file.

Table 23-4: ProFTPD Configuration Directives, proftpd.conf

Directive	Description
AccessGrantMsg *message*	Response message sent to an FTP client indicating the user has logged in or anonymous access has been granted. The magic cookie '**%u**' is replaced with the username specified by the client.Default: Dependent on login typeContext: server config, VirtualHost, Anonymous, Global
Allow ["from"] "all"|"none"|*host*|*network*[,*host*|*network*[,...]]	Used inside a Limit context to specify explicitly which hosts and/or networks have access to the commands or operations being limited. Used with Order and Deny to create access control rules. Default: Allow from all Context: Limit
AllowAll	Allows access to a Directory, Anonymous or Limit block Default: Default is to implicitly AllowAll, but not explicitly Context: Directory, Anonymous, Limit, .ftpaccess
AllowFilter *regular-expression*	Allows the configuration of a regular expression that must be matched for all commands sent to ProFTPD. Default: None Context: server config, VirtualHost, Anonymous, Global
AllowForeignAddress on|off	Allows clients to transmit foreign data connection addresses that do not match the client's address. Default: AllowForeignAddress off Context: server config, VirtualHost, Anonymous, Global
AllowGroup *group-expression*	List of groups allowed in a Limit block. Default: None Context: Limit
AllowUser *user-expression*	Users allowed access. Default: None Context: Limit
AnonRequirePassword *on|off*	Requires anonymous logins to enter a valid password that must match the password of the user that the anonymous daemon runs as. This is used to create guest accounts that function like anonymous logins but require a valid password. Default: AnonRequirePassword off Context: Anonymous
Anonymous *root-directory*	Create an anonymous FTP login, terminated by a matching /Anonymous directive. The root directory parameter is the directory **proftpd** first moves to and then **chroot** to, hiding the rest of the file system. Default: None Context: server config,VirtualHost
AuthGroupFile *path*	Alternate group's file with the same format as the system **/etc/group** file.

Table 23-4: ProFTPD Configuration Directives, proftpd.conf

Directive	Description
	Default: None Context: server config, VirtualHost, Global
AuthUserFile *path*	Alternate **passwd** file with the same format as the system **/etc/passwd** file. Default: None Context: server config,VirtualHost, Global
Bind *address*	Allows additional IP addresses to be bound to a main or VirtualHost configuration. Multiple Bind directives can be used to bind multiple addresses. Default: None Context: server config, VirtualHost
DefaultRoot *directory [group-expression]*	Default root directory assigned to user on login. The **group-expression** argument restricts the DefaultRoot directive to a group or set of groups. Default: DefaultRoot /Context: server config, VirtualHost, Global
Deny *["from"] "all"*|*"none"*|*host*|*network[,host*|*network[,...]]*	List of hosts and networks explicitly denied access to a given Limit context block. **all** indicates all hosts are denied access, and **none** indicates no hosts are explicitly denied. Default: None Context: Limit
DenyAll	Deny access to a directory, anonymous FTP, or Limit block. Default: None Context: Directory, Anonymous, Limit, .ftpaccess
DenyFilter *regular-expression*	Specifies a regular expression, which must *not* match any command. Default: None Context: server config, VirtualHost, Anonymous, Global
DenyUser *user-expression*	Users denied access within a Limit block. Default: None Context: Limit
Directory *pathname*	Directory-specific configuration. Used to create a block of directives that apply to the specified directory and its subdirectories. Default: None Context: server config, VirtualHost, Anonymous, Global
DisplayFirstChdir *filename*	Specifies the text file displayed to a user the first time he or she changes into a given directory during an FTP session.

Table 23-4: ProFTPD Configuration Directives, proftpd.conf

Directive	Description
	Default: None Context: server config, VirtualHost, Anonymous, Directory, Global
DisplayLogin ***filename***	Specifies the text file displayed to a user who logs in. Default: None Context: server config, VirtualHost, Anonymous, Global
Global	Global configuration block is used to create a set of configuration directives applied universally to both the main server configuration and all VirtualHost configurations. Default: None Context: server config, VirtualHost
Limit ***command|command-group [command2 ..]***	Access restrictions on FTP commands, within a given context. The **command-group** refers to groupings of commands as defined in the ProFTPD documentation. Default: None Context: server config, VirtualHost, Directory, Anonymous, Global, .ftpaccess
LsDefaultOptions ***"options string"***	Default options for directory listings (as in the **ls** command). Default: None Context: server config, VirtualHost, Global
MaxClients ***number| none message***	Maximum number of connected clients allowed. The message specified is displayed when a client is refused connection. Default: MaxClients none Context: server config, Anonymous, VirtualHost, Global
MaxLoginAttempts ***number***	Maximum number of times a client may attempt to log in to the server during a given connection. Default: MaxLoginAttempts 3 Context: server config, VirtualHost, Global
Order ***allow,deny|deny,allow***	Configures the order that Allow and Deny directives are checked inside of a Limit block. Default: Order allow, deny Context: Limit
PersistentPasswd on|off	When on, **proftpd**, during login, opens the system-wide **/etc/passwd**, **/etc/group** files, accessing them even during a **chroot** operation that changes the root directory. Default: Platform dependent Context: server config
RequireValidShell ***on|off***	Allow or deny logins not listed in **/etc/shells**. By default, **proftpd** disallows logins if the user's default shell is not

Table 23-4: ProFTPD Configuration Directives, proftpd.conf

Directive	**Description**
	listed in **/etc/shells**. Default: RequireValidShell on Context: server config, VirtualHost, Anonymous, Global
ScoreboardPath ***path***	Directory that holds **proftpd** runtime **Scoreboard** files. Default: ScoreboardPath /var/run Context: server config
ServerAdmin ***"admin-email-address"***	E-mail address of the server or virtual host administrator. Default: ServerAdmin root@[ServerName] Context: server config, VirtualHost
ServerType ***type-identifier***	The server daemon's operating mode, either inetd or standalone. Default: ServerType standalone Context: server config
TimeoutIdle ***seconds***	Maximum number of seconds proftpd allows clients to stay connected without any activity. Default: TimeoutIdle 600 Context: server config
Umask ***octal-mask***	Permissions applied to newly created file and directory within a given context. Default: None Context: server config, Anonymous, VirtualHost, Directory, Global, .ftpaccess
User ***userid***	The user the **proftpd** daemon runs as. Default: None Context: server config, VirtualHost, Anonymous, Global
UserAlias ***login-user userid***	Maps a login name used by a client to a user ID on the server. A client logging in as *login-user* is actually logged in as *user ID*. Often used inside an Anonymous block to allow specified *login-names* to perform an anonymous login. Default: None Context: server config, VirtualHost, Anonymous, Global
VirtualHost ***address***	Configuration directives that apply to a particular hostname or IP address. Often used with virtual servers that run on the same physical machine. The block is terminated with a /VirtualHost directive. By using the Port directive inside a VirtualHost block, creating a virtual server that uses the same address as the master server, but that listens on a separate TCP port, is possible. Default: None Context: server config

```
proftpd -c newfile.conf
```

Different kinds of directives exist. Many set values, such as MaxClients, which set the maximum number of clients, or NameServer, which sets the name of the FTP server. Others create blocks that can hold directives that apply to specific FTP server components. Block directives are entered in pairs: a beginning directive and a terminating directive. The terminating directive defines the end of the block and consists of the same name, beginning with a slash. Block directives take an argument that specifies the particular object to which the directives will apply. For the Directory block directive, you must specify a directory name to which it will apply. The <Directory *mydir*> block directive creates a block whose directives within it apply to the *mydir* directory. The block is terminated by a </Directory> directive. <Anonymous *ftp-dir*> configures the anonymous service for your FTP server. You need to specify the directory on your system used for your anonymous FTP service, such as **/var/ftp**. The block is terminated with the </Anonymous> directive. The <VirtualHost *hostaddress*> block directive is used to configure a specific virtual FTP server and must include the IP or the domain name address used for that server. </VirtualHost> is its terminating directive. Any Directives you place within this block are applied to that virtual FTP server. The <Limit *permission*> directive specifies the kind of access you want to limit. It takes as its argument one of several keywords indicating the kind of permission to be controlled: **WRITE** for write access, **READ** for read access, **STOR** for transfer access (uploading), and **LOGIN** to control user login.

A sample of the standard **proftpd.conf** file installed as part of the ProFTPD software package is shown here. Notice the default ServerType is standalone. If you want to use xinetd to run your server, you must change this entry to inetd. Detailed examples of **proftpd.conf** files, showing various anonymous FTP and virtual host configurations, can be found with the ProFTPD documentation, located in **/usr/doc**, and on the ProFPTD Web site at **www.proftpd.net**.

```
# This is a basic ProFTPD configuration file (rename it to
# 'proftpd.conf' for actual use.  It establishes a single server
# and a single anonymous login.  It assumes that you have a user/group
# "nobody" and "ftp" for normal operation and anon.

ServerName         "ProFTPD Default Installation"
ServerType         standalone
DefaultServer         on

# Port 21 is the standard FTP port.
Port            21
Umask         022
MaxInstances         30

# Set the user and group that the server normally runs at.
User            nobody
Group         nobody

# Normally, we want files to be overwriteable.
<Directory /*>
       AllowOverwrite      on
</Directory>

# A basic anonymous configuration, with one incoming directory.
<Anonymous ~ftp>
       User            ftp
```

```
  Group            ftp
  RequireValidShell      off
  MaxClients         10
  # We want clients to be able to login with "anonymous" as well as "ftp"
  UserAlias         anonymous ftp

  # We want 'welcome.msg' displayed at login, and '.message' displayed
  # in each newly chdired directory.
  DisplayLogin        welcome.msg
  DisplayFirstChdir      .message

  # Limit WRITE everywhere in the anonymous chroot except incoming
  <Directory *>
     <Limit WRITE>
         DenyAll
     </Limit>
  </Directory>

  <Directory incoming>
     <Limit WRITE>
            AllowAll
     </Limit>
     <Limit READ>
            DenyAll
     </Limit>
  </Directory>

</Anonymous>
```

Anonymous Access

You use the Anonymous configuration directive to create an anonymous configuration block in which you can place directives that configure your anonymous FTP service. The directive includes the directory on your system used for the anonymous FTP service. The ProFTPD daemon executes a **chroot** operation on this directory, making it the root directory for the remote user accessing the service. By default, anonymous logins are supported, expecting users to enter their e-mail address as a password. You can modify an anonymous configuration to construct more controlled anonymous services, such as guest logins and required passwords.

Note For ProFTPD, your anonymous FTP directory does not require any system files. Before ProFTPD executes a **chroot** operation, hiding the rest of the system from the directory, it accesses and keeps open any needed system files outside the directory.

The following example shows a standard anonymous FTP configuration. The initial Anonymous directive specifies **/var/ftp** as the anonymous FTP home directory. The User directive specifies the user that the Anonymous FTP daemon will run as, and Group indicates its group. In both cases, FTP, the standard username, is used on most systems for anonymous FTP. A Directory directive with the * file matching character then defines a Directory block that applies to all directories and files in **/var/ftp**. The * symbol matches on all filenames and directories. Within the Directory directive is a Limit directive that you use to place controls on a directory. The directive takes several arguments including **READ** for read access and **WRITE** for write access. In this example, the Limit directive places restrictions on the write capabilities of users. Within the Limit directive, the DenyAll directive denies write permission, preventing users from creating or deleting files and effectively giving them only read access. A second Directory directive creates an exception to this rule for the incoming directory. An incoming directory is usually set up on FTP sites to let users upload files. For this directory, the first Limit directive prevents both **READ** and **WRITE** access by users with its DenyAll directive, effectively preventing users from deleting or reading files here. The second Limit directive lets users upload files, however, permitting transfers only (**STOR**) with the AllowAll directive.

One important directive for anonymous FTP configurations is the RequireValidShell. By default, the FTP daemon first checks to see if the remote user is attempting to log in using a valid shell, such as the BASH shell or the C shell. The FTP daemon obtains the list of valid shells from the **/etc/shells** file. If the remote user does not have a valid shell, a connection is denied. You can turn off the check using the RequireValidShell directive and the **off** option. The remote user can then log in using any kind of shell.

```
<Anonymous /var/ftp>
    User ftp
    Group ftp
    UserAlias anonymous ftp
    RequireValidShell off
 <Directory *>
       <Limit WRITE>
               DenyAll
       </Limit>
    </Directory>
     # The only command allowed in incoming is STOR
     # (transfer file from client to server)
    <Directory incoming>
       <Limit READ WRITE>
               DenyAll
       </Limit>
       <Limit STOR>
               AllowAll
       </Limit>
    </Directory>
 </Anonymous>
```

Recall that FTP was originally designed to let a remote user connect to an account of his or her own on the system. Users can log in to different accounts on your system using the FTP service. Anonymous users are restricted to the anonymous user account. However, you can create other users and their home directories that also function as anonymous FTP accounts with the same restrictions. Such accounts are known as *guest accounts.* Remote users are required to know the username and, usually, the password. Once connected, they only have read access to that account's files; the rest of the file system is hidden from them. In effect, you are creating a separate anonymous FTP site at the same location with more restricted access.

To create a guest account, first create a user and the home directory for it. You then create an Anonymous block in the **proftpd.conf** file for that account. The Anonymous directive includes the home directory of the guest user you create. You can specify this directory with a ~ for the path and the directory name, usually the same as the username. Within the Anonymous block, you use the User and Group directives to specify the user and group name for the user account. Set the AnonRequirePassword directive to **on** if you want remote users to provide a password. A UserAlias directive defines aliases for the username. A remote user can use either the alias or the original username to log in. You then enter the remaining directives for controlling access to the files and directories in the account's home directory. An example showing the initial directives is listed here. The User directive specifies the user as **myproject**. The home directory is **~myproject**, which usually evaluates to **/var/myproject**. The UserAlias lets remote users log in either with the name **myproject** or **mydesert**.

```
<Anonymous ~myproject>
    User myproject
    Group other
    UserAlias mydesert myproject
    AnonRequirePassword on
    <Directory *>
```

You could just as easily create an account that requires no password, letting users enter their e-mail addresses instead. The following example configures an anonymous user named **mypics**. A password isn't required and neither is a valid shell. The remote user still needs to know the username, in this case **mypics**.

```
<Anonymous /var/mypics>
    AnonRequirePassword off
    User mypics
    Group nobody
    RequireValidShell off
    <Directory *>
```

The following example provides a more generic kind of guest login. The username is **guest** with the home directory located at **~guest**. Remote users are required to know the password for the guest account. The first Limit directive lets all users log in. The second Limit directive allows write access from users on a specific network, as indicated by the network IP address, and denies write access by any others.

```
<Anonymous ~guest>
  User            guest
  Group            nobody
  AnonRequirePassword      on

  <Limit LOGIN>
        AllowAll
  </Limit>

  # Deny write access from all except trusted hosts.
  <Limit WRITE>
        Order          allow,deny
        Allow          from 10.0.0.
        Deny          from all
  </Limit>

</Anonymous>
```

Virtual FTP Servers

The ProFTPD daemon can manage more than one FTP site at once. Using a VirtualHost directive in the **proftpd.conf** file, you can create an independent set of directives that configure a separate FTP server. The VirtualHost directive is usually used to configure virtual servers as FTP sites. You can configure your system to support more than one IP address. The extra IP addresses can be used for virtual servers, not independent machines. You can use such an extra IP address to set up a virtual FTP server, giving you another FTP site on the same system. This added server would use the extra IP address as its own. Remote users could access it using that IP address, instead of the system's main IP address. Because such an FTP server is not running independently on a separate machine but is, instead, on the same machine, it is known as a *virtual FTP server* or *virtual host.* This feature lets you run what appears to others as several different FTP servers on one machine. When a remote user uses the virtual FTP server's IP address to access it, the ProFTPD daemon detects that request and operates as the FTP service for that site. ProFTPD can handle a great many virtual FTP sites at the same time on a single machine.

Note Given its configuration capabilities, you can also tailor any of the virtual FTP sites to specific roles, such as a guest site, an anonymous site for a particular group, or an anonymous site for a particular user.

You configure a virtual FTP server by entering a <VirtualHost> directive for it in your **proftpd.conf** file. Such an entry begins with the VirtualHost directive and the IP address, and ends with a terminating VirtualHost directive, </VirtualHost>. Any directives placed within these are applied to the virtual host. For anonymous or guest sites, add Anonymous and Guest directives. You can even add Directory directives for specific directories. With the Port directive on a standalone configuration, you can create a virtual host that operates on the same system but connects on a different port.

```
<VirtualHost 10.0.0.1>
      ServerName "My virtual FTP server"
</VirtualHost>
```

Xinetd and standalone configurations handle virtual hosts differently. Xinetd detects a request for a virtual host, and then hands it off to an FTP daemon. The FTP daemon then examines the address and port specified in the request and processes the request for the appropriate virtual host. In the standalone configuration, the FTP daemon continually listens for requests on all specified ports and generates child processes to handle ones for different virtual hosts as they come in. In the standalone configuration, ProFTPD can support a great many virtual hosts at the same time.

The following example shows a sample configuration of a virtual FTP host. The VirtualHost directives use domain name addresses for their arguments. When a domain name address is used, it must be associated with an IP address in the network's domain name server. The IP address, in turn, has to reference the machine on which the ProFTPD daemon is running. On the **ftp.mypics.com** virtual FTP server, an anonymous guest account named **robpics** is configured that requires a password to log in. An anonymous FTP account is also configured that uses the home directory **/var/ftp/virtual/pics**.

```
<VirtualHost ftp.mypics.com>

  ServerName         "Mypics FTP Server"
  MaxClients         10
  MaxLoginAttempts      1
  DeferWelcome         on
  <Anonymous ~robpics>
    User         robpics
    Group         robpics
    AnonRequirePassword      on

  <Anonymous /var/ftp/virtual/pics>
    User         ftp
    Group         ftp
    UserAlias         anonymous ftp
  </Anonymous>
</VirtualHost>
```

Chapter 24: Red Hat Web Servers: Apache and Tux

Red Hat provides several Web servers for use on your system. The primary Web server is Apache, which has almost become the standard Web server for Linux. It is a very powerful, stable, and fairly easy to configure system. Tux is smaller, but very fast, and can handle Web data that does not change with great efficiency. Red Hat provides default configurations for the Web servers, making them usable as soon as they are installed.

Tux

Tux is a static content Web server designed to be run very fast from within the Linux kernel. In effect it runs in kernel space, making response times much faster than standard user-space Web servers like Apache. As a kernel-space server, Tux can handle static content such as images very efficiently. At the same time it can coordinate with a user-space Web server, like Apache, to provide the dynamic content, like CGI programs. Tux can even make use of a cache to hold previously generated dynamic content, using it as if it were static. The ability to coordinate with a user-space Web server lets you use Tux as your primary Web server. Anything that Tux cannot handle, it will pass off to the user-space Web server.

Note Tux is freely distributed under the GNU Public License and is included with Red Hat distributions.

The Tux configuration file is located in **/proc/sys/net/tux**. Here you enter parameters such as serverport, max_doc_size, and logfile (check the Tux reference manual at **www.redhat.com/support/manuals** for a detailed listings). Red Hat defaults are already entered. serverport, clientport, and documentroot are required parameters that must be set. serverport is the port Tux will use-80 if it is the primary Web server. clientport is the port used by the user-space Web server Tux coordinates with, like Apache. documentroot specifies the root directory for your Web documents (**/var/www/html** on Red Hat).

Ideally, Tux is run as the primary Web server and Apache as the secondary Web server. To configure Apache to run with Tux, the port entry in the Apache **httpd.conf** file needs to be changed from 80 to 8080.

```
Port 8080
```

You can start, stop, and restart the server with the **/etc/rd.d/init.d/tux** command. Several parameters like DOCROOT can be specified as arguments to this tux command. You can enter them in the **/etc/sysconfig/tux** file.

Note You can also run Tux as an FTP server. In the **/proc/sys/net/tux** directory, you change the contents of the files serverport to 21, application_protocol to 1, and nonagle to 0, and then restart Tux. Use the **generatetuxlist** command in the document root directory to generate FTP directory listings.

Apache Web Server

The Apache Web server is a full-featured free HTTP (Web) server developed and maintained by the Apache Server Project. The aim of the project is to provide a reliable, efficient, and easily extensible Web server, with free open-source code. The server software includes the server daemon, configuration files, management tools, and documentation. The Apache Server Project is maintained by a core group of volunteer programmers and supported by a great many contributors worldwide. The Apache Server Project is one of several projects currently supported by the Apache Software Foundation (formerly known as the Apache Group). This nonprofit organization provides financial, legal, and organizational support for various Apache open-source software projects, including the Apache HTTPD Server, Java Apache, Jakarta, and XML-Apache. The Web site for the Apache Software Foundation is at **www.apache.org**. Table 24-1 lists various Apache-related Web sites.

Apache was originally based on the NCSA Web server developed at the National Center for Supercomputing Applications, University of Illinois, Urbana- Champaign. Apache has since emerged as a server in its own right and has become one of the most popular Web servers in use. Although originally developed for Linux and Unix systems, Apache has become a cross-platform application with Windows and OS2 versions. Apache provides online support and documentation for its Web server at **httpd.apache.org**. An HTML-based manual is also provided with the server installation. You can use the Apache configuration tool, installed with Red Hat, to help configure your Apache server easily. It operates on any X Windows window manager, including Gnome and KDE. In addition, you can use the Comanche configuration tool. Webmin and Linuxconf also provide Apache configuration support.

Table 24-1: Apache-Related Web Sites

Web Site	**Description**
www.apache.org	Apache Software Foundation
httpd.apache.org	Apache HTTP Server Project
java.apache.org	Java Apache Project
jakarta.apache.org	Jakarta Apache Project
gui.apache.org	Apache GUI Project
www.comanche.org	Comanche (Configuration Manager for Apache)
www.apache-ssl.org	Apache-SSL server (Secure Socket Layer)
www.openssl.org	OpenSSL project
www.modssl.org	The SSL module (mod_ssl) project to add SSL encryption to an Apache Web server

Other Web servers available for Linux include the Red Hat Secure Server (**www.redhat.com**), Apache-SSL (**www.apache-ssl.org**), Stronghold (**www.c2.net**), and Netscape Enterprise Server (**home.netscape.com**). Apache-SSL is an encrypting Web server based on Apache and OpenSSL (**www.openssl.org**). Stronghold is a commercial version of the Apache Web server featuring improved security and administration tools. You can also use the original NCSA Web server, though it is no longer supported (**hoohoo.ncsa.uiuc.edu**). AOLserver is America Online's Web server that is now available under the GPL license (**www.aolserver.com**).

Java: Jakarta and Apache-Java

The Java Apache Project develops open-source Java software and has its Web site located at **java.apache.org**. Currently, the Java Apache Project supports numerous projects, including JServ, JSSI, JMeter, and mod_java, among others. The Apache JServ Project has developed a Java servlet engine compliant with the JavaSoft Java Servlet API's 2.0 specification. The Apache JSSI project has developed a Java servlet for dynamic servlet output from HTML files through the <SERVLET> tag as designated by the JavaSoft Java Web Server. JMeter is a Java desktop application to test performance of server resources, such as servlets and CGI scripts. The MOD_JAVA project has developed a mod_java extension module for Apache Web servers that allows Apache modules to be written in Java instead of in C. It functions much like mod_perl (which allows modules to be written in Perl).

Jakarta is an Apache project to develop server-side Java capabilities on Linux. Jakarta's main product, called Tomcat*,* is an open-source implementation of the Java Servlet 2.2 and JavaServer Pages 1.1 specifications. Tomcat is designed for use in Apache servers. The Jakarta Web site is at **jakarta.apache.org**.

Linux Distribution Apache Installations

Red Hat provides you with the option of installing the Apache Web server during your initial installation of your Linux system. All the necessary directories and configuration files are automatically generated for you. Then, whenever you run Linux, your system is already a fully functional Web site. Every time you start your system, the Web server will also start up, running continuously. On Red Hat systems, the directory reserved for your Web site data files is **/var/www/html**. Place your Web pages in this directory or in any subdirectories. Your system is already configured to operate as a Web server. All you need to do is perform any needed network server configurations, and then designate the files and directories open to remote users. You needn't do anything else. Once your Web site is connected to a network, remote users can access it.

The Web server installed on Red Hat systems sets up your Web site in the **/var/www** directory. It also sets up several directories for managing the site. The **/var/www/cgi-bin** directory holds the CGI scripts, and **/var/www/html/manual** holds the Apache manual in HTML format. You can use your browser to examine it. Your Web pages are to be placed in the **/var/www/html** directory. Place your Web site home page there. Your configuration files are located in a different directory, **/etc/httpd/conf**. Table 24-2 lists the various Apache Web server directories and configuration files.

Table 24-2: Apache Web Server Files and Directories (RPM Installation)

Web Site Directories	**Description**
/var/www	Directory for Apache Web site files on your Red Hat system

Table 24-2: Apache Web Server Files and Directories (RPM Installation)

Web Site Directories	**Description**
/var/www/html	Web site Web files
/var/www/cgi-bin	CGI program files
/var/www/html/manual	Apache Web server manual
Configuration Files	
.htaccess	Directory-based configuration files. An **.htaccess** file holds directives to control access to files within the directory in which it is located
/etc/httpd/conf	Directory for Apache Web server configuration files
/etc/httpd/conf/httpd.conf	Apache Web server configuration file
Startup Scripts	
/etc/rc.d/init.d/httpd	Startup script for Web server daemon
/etc/rc.d/rc3.d/S85httpd	Link in runlevel 3 directory (**/etc/rc3.d**) to the httpd startup script in the **/etc/rc.d/init.d** directory
Application Files	
/usr/sbin	Location of the Apache Web server program file and utilities
/usr/share/doc/	Apache Web server documentation
/var/log/http	Location of Apache log files

To upgrade your Apache server, either use the Red Hat upgrade tool, or look for recent Apache update files at your Linux distribution's FTP sites. There, you can download RPM packages containing the latest version of the Apache Web server, specially configured for your Linux distribution. Then use the **rpm** command with the **-Uvh** options to install the upgrade (the **-U** option specifies an upgrade option).

```
rpm -Uvh apache-1.3.12-20.i386.rpm
rpm -Uvh apache-docs-1.3.12-20.i386.rpm
```

Alternatively, you can download the source code version for the latest Apache Web server directly from Apache and compile it on your system. You must decompress the file and extract the archive. Many of the same directories are created, with added ones for the source code. The server package includes installation instructions for creating your server directories and compiling your software. Make sure the configuration files are set up and installed.

Note If you are installing Apache from the source code, notice that versions of the configuration files ending with the extension **.conf-dist** are provided. You have to make copies of these configuration files with the same prefix, but only with the extension **.conf** to set up a default configuration. The Web server reads configuration information only from files with a **.conf** extension.

Apache Web Server 2.0

Red Hat 7.1 still uses Apache version 1.3, which is described here. Recently Apache released 2.0, which is designed to be less dependent on Linux/Unix systems. Most directives and features for Apache 1.3 still work on Apache 2.0. However, Apache 2.0 has introduced a new architecture that uses Multi-Processing Modules (MPMs), which are designed to customize Apache to different operating systems. A Linux system would use the threaded MPM, whereas Windows would use the mpm_winnt MPM. To maintain compatibility with Apache 1.3 configurations, you would use the prefork MPM.

Apache 2.0 has adopted a much more modular architecture than 1.3. Many directives that once resided in the Apache core are now placed in respective modules and MPMs. With this modular design, several directives have been dropped, such as ServerType. Such directives deprecated in Apache 2.0 are noted in Table 24-3.

Table 24-3: Apache Modules (Apache 2.0)

Module	**Description**
mod_access	Accesses control based on client hostname or IP address
mod_actions	Executes CGI scripts based on media type or request method
mod_alias	Maps different parts of the host file system in the document tree, and URL redirection
mod_asi	Sends files that contain their own HTTP headers
mod_auth	User authentication using text files
mod_auth_anon	Anonymous user access to authenticated areas
mod_auth_db	User authentication using Berkeley DB files
mod_auth_dbm	User authentication using DBM files
mod_auth_digest	MD5 authentication
mod_autoindex	Automatic directory listings
mod_cern_meta	Support for HTTP header metafiles
mod_cgi	Invokes CGI scripts
mod_cgid	Invokes CGI scripts using an external daemon
mod_charset_lite	Configures character set translation
mod_dav	Class 1,2 WebDAV HTTP extensions
mod_dir	Basic directory handling
mod_env	Passes environments to CGI scripts
mod_example	Demonstrates Apache API
mod_expires	Applies Expires: headers to resources
mod_ext_filter	Filters output with external programs
mod_file_cache	Caches files in memory for faster serving
mod_headers	Adds arbitrary HTTP headers to resources
mod_imap	The image-map file handler
mod_include	Server-parsed documents

Table 24-3: Apache Modules (Apache 2.0)

Module	Description
mod_info	Server configuration information
mod_isapi	Windows ISAPI Extension support
mod_log_config	User-configurable logging replacement for mod_log_common
mod_mime	Determines document types using file extensions
mod_mime_magic	Determines document types using "magic numbers"
mod_negotiation	Content negotiation
mod_proxy	Caches proxy abilities
mod_rewrite	Powerful URI-to-filename mapping using regular expressions
mod_setenvif	Sets environment variables based on client information
mod_so	Support for loading modules at runtime
mod_spelling	Automatically corrects minor typos in URLs
mod_status	Server status display
mod_unique_id	Generates unique request identifier for every request
mod_userdir	User home directories
mod_usertrack	User tracking using Cookies (replacement for mod_cookies.c)
mod_vhost_alias	Support for dynamically configured mass virtual hosting

Note A selected MPM is usually integrated into Apache when it is compiled. Future Red Hat distributions of Apache should use the Linux/Unix default MPM, which is named "threaded."

Starting and Stopping the Web Server

On most systems, Apache is installed as a stand-alone server, continually running. As noted in Chapter 22, in the discussion of init scripts, your system automatically starts up the Web server daemon, invoking it whenever you start your system. On Red Hat systems, a startup script for the Web server called httpd is in the **/etc/rc.d/init.d** directory. Symbolic links through which this script is run are located in corresponding runlevel directories. You will usually find the **S85httpd** link to **/etc/rc.d/init.d/httpd** in the runlevel 3 and 5 directories, **/etc/rc.d/rc3.d** and **/etc/rc.d/rc5.d**. You can use the **chkconfig** command or the System V Init Editor to set the runlevels at which the httpd server will start, creating links in appropriate runlevel directories. The following command will set up the Web server (httpd) to start up at runlevels 3 and 5 (see Chapter 22 for more details on runlevels).

```
chkconfig --level 35 httpd on
```

You can also use **service** command to start and stop the httpd server manually. This may be helpful when you are testing or modifying your server. The httpd script with the **start** option starts the server, the **stop** option stops it, and **restart** will restart it. Simply killing the Web process directly is not advisable.

```
service httpd restart
```

Apache also provides a control tool called **apachectl** (Apache control) for managing your Web server. With **apachectl**, you can start, stop, and restart the server from the command line. **apachectl** takes several arguments: **start** to start the server, **stop** to stop it, **restart** to shut down and restart the server, and **graceful** to shut down and restart gracefully. In addition, you can use **apachectl** to check the syntax of your configuration files with the **config** argument. You can also use **apachectl** as a system startup file for your server in the **/etc/rc.d** directory.

Remember, httpd is a script that calls the actual **httpd** daemon. You could call the daemon directly using its full pathname. This daemon has several options. The **-d** option enables you to specify a directory for the httpd program if it is different from the default directory. With the **-f** option, you can specify a configuration file different from **httpd.conf**. The **-v** option displays the version.

```
/usr/sbin/httpd -v
```

To check your Web server, start your Web browser and enter the Internet domain name address of your system. For the system **turtle.mytrek.com,** the user enters **http://turtle.mytrek.com**. This should display the home page you placed in your Web root directory. A simple way to do this is to use Lynx, the command line Web browser. Start Lynx**,** and then press **g** to open a line where you can enter a URL for your own system. Then Lynx displays your Web site's home page. Be sure to place an **index.html** file in the **/var/www/html** directory first.

Once you have your server running, you can check its performance with the **ab** benchmarking tool, also provided by Apache. **ab** shows you how many requests at a time your server can handle. Options include **-v,** which enables you to control the level of detail displayed, **-n,** which specifies the number of requests to handle (default is 1), and **-t,** which specifies a time limit.

Note Currently there is no support for running Apache under xinetd. In Apache 2.0, such support is determined by choosing an MPM designed to run on xinetd. Currently there are none.

Apache Configuration and Directives

Configuration directives are placed in the **httpd.conf** configuration file. An documented version of the **httpd.conf** configuration file for Red Hat is installed automatically in **/etc/httpd**. It is strongly recommended that you consult this file on your system. It contains detailed documentation and default entries for Apache directives.

Any of the directives in the main configuration files can be overridden on a per-directory basis using an **.htaccess** file located within a directory. Although originally designed only for access directives, the **.htaccess** file can also hold any resource directives, enabling you to tailor how Web pages are displayed in a particular directory. You can configure access to .**htaccess** files in the **httpd.conf** file.

Note With Apache version 1.3.4, all configuration directives are placed in one file, the **httpd.conf** file. Older versions used two other files, the **srm.conf** and **access.conf** files. The **srm.conf** file handled document specifications, configuring file types and locations.

> The **access.conf** file was designed to hold directives that control access to Web site directories and files. Though you will still find these files on current Apache versions, they will be empty.

Apache configuration operations take the form of directives entered into the Apache configuration files. With these directives, you can enter basic configuration information, such as your server name, or perform more complex operations, such as implementing virtual hosts. The design is flexible enough to enable you to define configuration features for particular directories and different virtual hosts. Apache has a variety of different directives performing operations as diverse as controlling directory access, assigning file icon formats, and creating log files. Most directives set values such as **DirectoryRoot**, which holds the root directory for the server's Web pages, or Port, which holds the port on the system that the server listens on for requests. Table 24-3 provides a listing of the more commonly used Apache directives. The syntax for a simple directive is shown here.

```
directive option option …
```

Certain directives create blocks able to hold directives that apply to specific server components (also referred to as sectional directives). For example, the **Directory** directive is used to define a block within which you place directives that apply only to a particular directory. Block directives are entered in pairs: a beginning directive and a terminating directive. The terminating directive defines the end of the block and consists of the same name beginning with a slash. Block directives take an argument that specifies the particular object to which the directives apply. For the **Directory** block directive, you must specify a directory name to which it will apply. The **<Directory *mydir*>** block directive creates a block whose directives within it apply to the *mydir* directory. The block is terminated by a **</Directory>** directive. The **<VirtualHost *hostaddress*>** block directive is used to configure a specific virtual Web server and must include the IP or domain name address used for that server. **</VirtualHost>** is its terminating directive. Any directives you place within this block are applied to that virtual Web server. The **<Limit *method*>** directive specifies the kind of access method, such as GET or POST, you want to limit. The access control directives located within the block list the controls you are placing on those methods. The syntax for a block directive is as follows:

```
<block-directive option … >
 directive option …
 directive option …
</block-directive>
```

Usually, directives are placed in one of the main configuration files. **Directory** directives in those files can be used to configure a particular directory. However, Apache also makes use of directory-based configuration files. Any directory may have its own **.htaccess** file that holds directives to configure only that directory. If your site has many directories, or if any directories have special configuration needs, you can place their configuration directives in their **.htaccess** files, instead of filling the main configuration file with specific **Directory** directives for each one. You can control what directives in an **.htaccess** file take precedence over those in the main configuration files. If your site allows user- or client-controlled directories, you may want to carefully monitor or disable the use of **.htaccess** files in them. (It is possible for directives in an **.htaccess** file to override those in the standard configuration files unless disabled with **AllowOverride** directives.)

Much of the power and flexibility of the Apache Web server comes from its use of modules to extend its capabilities. Apache is implemented with a core set of directives. Modules can be created that hold definitions of other directives. They can be loaded into Apache, enabling you to use those directives for your server. A standard set of modules is included with the Apache distribution, though you can download others and even create your own. For example, the mod_autoindex module holds the directives for automatically indexing directories (as described in the following section). The mod_mime module holds the MIME type and handler directives. Modules are loaded with the **LoadModule** directive. On Red Hat Linux, you can find **LoadModule** directives in the **httpd.conf** configuration file for most of the standard modules.

```
LoadModule mime_module modules/mod_mime.so
```

The apxs application provided with the Apache package can be used to build Apache extension modules. With the apxs application, you can compile Apache module source code in C and create dynamically shared objects that can be loaded with the **LoadModule** directive. The apxs application requires that the mod_so module be part of your Apache application. It includes extensive options such as **-n** to specify the module name, **-a** to add an entry for it in the **httpd.conf** file, and **-i** to install the module on your Web server.

You can find a complete listing of Apache Web configuration directives at the Apache Web site, **httpd.apache.org**, and in the Apache manual located in your site's Web site root directory. On Red Hat systems, this is located at **/var/www/manual**. Many of the more commonly used directives are listed in Table 24-4.

Table 24-4: Apache Log Field Specifiers

Format Specifier	**Description**
%a	Remote IP address
%A	Local IP address
%b	Bytes sent, excluding HTTP headers
%{*variable*}**e:**	The contents of the environment *variable*
%f	Filename
%h	Remote host
%l	Remote logname (from **identd**, if supplied)
%m	The request method
%P	The process ID of the child that serviced the request
%r	First line of request
%s	Status
%t	Time, in Common Log Format time format (standard English format)
%u	Remote user (from auth; may be bogus if return status (**%s**) is 401)
%U	The URL path requested
%v	The canonical ServerName of the server serving the request

Server Configuration

Certain directives are used to configure your server's overall operations. These directives are placed in the **httpd.conf** configuration file. Some require pathnames, whereas others only need to be turned on or off with the keywords **on** and **off**. Apache provides a default **httpd.conf** configuration file. The **httpd.conf** file already contains these directives. Some are commented out with a preceding # symbol. You can activate a directive by removing its # sign. Many of the entries are preceded by comments explaining their purpose. The following is an example of the **ServerAdmin** directive used to set the address where users can send mail for administrative issues. You replace the you@your.address entry with the address you want to use to receive system administration mail. On Red Hat systems, this is set to root@localhost.

```
# ServerAdmin: Your address, where problems should be e-mailed.
ServerAdmin you@your.address
```

Some directives require specific information about your system. For example, **ServerName** holds the hostname for your Web server. Specifying a hostname is important to avoid unnecessary DNS lookup failures that can hang your server. Notice the entry is commented with a preceding #. Simply remove the # and type your Web server's hostname in place of **new.host.name**.

```
# ServerName allows you to set a hostname which is sent
# back to clients for your server if it's different than the
# one the program would get (i.e. use
# "www" instead of the host's real name).

#ServerName localhost
```

On Red Hat systems, entries have already been made for the standard Web server installation using **/var/www** as your Web site directory. You can tailor your Web site to your own needs by changing the appropriate directives. The **DocumentRoot** directive determines the home directory for your Web pages. The **ServerRoot** directive specifies where your Web server configuration, error, and log files are kept.

```
DocumentRoot /var/www/html
ServerRoot /etc/httpd
```

The **MaxClients** directive sets the maximum number of clients that can connect to your server at the same time.

```
MaxClients 150
```

Directory-Level Configuration: .htaccess and <Directory>

One of the most flexible aspects of Apache is its ability to configure individual directories. With the **Directory** directive, you can define a block of directives that apply only to a particular directory. Such a directive can be placed in the **httpd.conf** or **access.conf** configuration file. You can also use an **.htaccess** file within a particular directory to hold configuration directives. Those directives are then applied only to that directory. The name ".htaccess" is actually set with the **AccessFileName** directive. You can change this if you want.

```
AccessFileName .htaccess
```

A Directory block begins with a **<Directory** ***pathname*>** directive, where *pathname* is the directory to be configured. The ending directive uses the same **<>** symbols, but with a slash preceding the term "Directory": **</Directory>**. Directives placed within this block apply only to the specified directory. The following example denies access to only the **mypics** directory by requests from **www.myvids.com**.

```
<Directory /var/www/html/mypics>
 Order Deny,Allow
 Deny from www.myvids.com
</Directory>
```

With the **Options** directive, you can enable certain features in a directory, such as the use of symbolic links, automatic indexing, execution of CGI scripts, and content negotiation. The default is the All option, which turns on all features except content negotiation (Multiviews). The following example enables automatic indexing (Indexes), symbolic links (FollowSymLinks), and content negotiation (Multiviews).

```
Options Indexes FollowSymLinks Multiviews
```

Configurations made by directives in main configuration files or in upper-level directories are inherited by lower-level directories. Directives for a particular directory held in **.htaccess** files and Directory blocks can be allowed to override those configurations. This capability can be controlled by the **AllowOverride** directive. With the "all" argument, **.htaccess** files can override any previous configurations. The "none" argument disallows overrides, effectively disabling the **.htaccess** file. You can further control the override of specific groups of directives. AuthConfig enables use of authorization directives, FileInfo is for type directives, Indexes is for indexing directives, Limit is for access control directives, and Options is for the **options** directive.

```
AllowOverride all
```

Access Control

With access control directives, such as **allow** and **deny**, you can control access to your Web site by remote users and hosts. The **allow** directive followed by a list of hostnames restricts access to only those hosts. The **deny** directive with a list of hostnames denies access by those systems. The argument "all" applies the directive to all hosts. The **order** directive specifies in what order the access control directives are to be applied. Other access control directives, such as **require,** can establish authentication controls, requiring users to log in. The access control directives can be used globally to control access to the entire site or placed within **Directory** directives to control access to individual directives. In the following example, all users are allowed access:

```
order allow,deny
allow from all
```

You can further qualify access control directives by limiting them to certain HTML access methods. HTML access methods are ways a browser interacts with your Web site. For example, a browser could get information from a page (GET) or send information through it (POST). You can control such access methods using the **<Limit>** directive. **Limit** takes as its argument a list of access methods to be controlled. The directive then pairs with a **</Limit>** directive to define a Limit block within which you can place access control directives. These directives only apply to the specified access methods. You can place such Limit blocks with a Directory block to set up controls of access methods for a specific directory. On the Red Hat distribution, the following Directory block in the **/etc/config/httpd.conf** file controls access methods for your Web site's home directory, **/var/www/html**.

```
# This should be changed to whatever you set DocumentRoot to.
<Directory /var/www/html>
Options Indexes FollowSymLinks
AllowOverride All
<Limit GET>
order allow,deny
allow from all
</Limit>
</Directory>
```

Controls are inherited from upper-level directories to lower-level ones. If you want to control access strictly on a per-directory basis to your entire Web site, you can use the following entry to deny access to all users. Then, in individual directories, you can allow access to certain users, groups, or hosts.

```
<Directory /var/www/html>
 Order Deny,Allow
 Deny from All
 </Directory>
```

URL Pathnames

Certain directives can modify or complete pathname segments of a URL used to access your site. The pathname segment of the URL specifies a particular directory or Web page on your site. Directives enable you to alias or redirect pathnames, as well as to select a default Web page. With the **Alias** directive, you can let users access resources located in other parts of your system, on other file systems, or on other Web sites. An alias can use a URL for sites on the Internet, instead of a pathname for a directory on your system. With the **Redirect** directive, you can redirect a user to another site.

```
Alias /mytrain /home/dylan/trainproj
Redirect /mycars http://www.myautos.com/mycars
```

If Apache is given only a directory to access, rather than a specific Web page, it looks for an index Web page located in that directory and displays it. The possible names for a default Web page are listed by the **DirectoryIndex** directive. The name usually used is **index.html**, but you can add others. The standard names are shown here. When Apache is given only a Web directory to access, it looks for and displays the **index.html** Web page located in it.

```
DirectoryIndex index.html index.shtml index.cgi
```

Apache also enables a user to maintain Web pages located in a special subdirectory in the user's home directory, rather than in the main Web site directory. Using a ~ followed by the user name accesses this directory. The name of this directory is specified with the **UserDir** directive. The default name is **public_html**, as shown here. The site

turtle.mytrek.com/~dylan accesses the directory **turtle.mytrek.com/home/dylan/public_html** on the host **turtle.mytrek.com**.

```
UserDir public_html
```

MIME Types

When a browser accesses Web pages on a Web site, it is often accessing many different kinds of objects, including HTML files, picture or sound files, and script files. To display these objects correctly, the browser must have some indication of what kind of objects they are. A JPEG picture file is handled differently from a simple text file. The server provides this type information in the form of MIME types (see Chapter 14). MIME types are the same types used for sending attached files through Internet mailers, such as Pine. Each kind of object is associated with a given MIME type. Provided with the MIME type, the browser can correctly handle and display the object.

The MIME protocol associates a certain type with files of a given extension. For example, files with a **.jpg** extension would have the MIME type image/jpeg. The **TypesConfig** directive holds the location of the **mime.types** file, which lists all the MIME types and their associated file extensions. **DefaultType** is the default MIME type for any file whose type cannot be determined. **AddType** enables you to modify the **mime.type** types list without editing the MIME file.

```
TypesConfig /etc/mime.types
DefaultType text/plain
```

Other type directives are used to specify actions to be taken on certain documents. **AddEncoding** lets browsers decompress compressed files on the fly. **AddHandler** maps file extensions to actions, and **AddLanguage** enables you to specify the language for a document. The following example marks filenames with the **.gz** extension as gzip-encoded files and files with the **.fr** extension as French language files:

```
AddEncoding x-gzip gz
AddLanguage fr .fr
```

A Web server can display and execute many different types of files and programs. Not all Web browsers are able to display all those files, though. Older browsers are the most limited. Some browsers, such as Lynx, are not designed to display even simple graphics. To allow a Web browser to display a page, the server negotiates with it to determine the type of files it can handle. To enable such negotiation, you need to enable the multiviews option.

```
Option multiviews
```

CGI Files

Common Gateway Interface (CGI) files are programs that can be executed by Web browsers accessing your site. CGI files are usually initiated by Web pages that execute the program as part of the content they display. Traditionally, CGI programs were placed in a directory called **cgi-bin** and could only be executed if they resided in such a special directory. Usually, only one **cgi-bin** directory exists per Web site. Red Hat systems set up a **cgi-bin** directory in the **/var/www** directory, **/var/www/cgi-bin**. Here, you place any CGI programs that can be executed on your Web site. The **ScriptAlias** directive specifies an alias for your **cgi-bin** directory. Any Web pages or browsers can use the alias to reference this directory.

```
ScriptAlias /cgi-bin/ /var/www/cgi-bin/
```

If you want to execute CGI programs that reside anywhere on your Web site, you can specify that files with a **.cgi** extension are treated as executable CGI programs. You do this with the **AddHandler** directive. This directive applies certain handlers to files of a given type. The handler directive to do this is included in the default **httpd.conf** file, provided with the Apache source code files, though commented out. You can remove the comment symbol (#) to enable it.

```
AddHandler cgi-script cgi
```

Automatic Directory Indexing

When given a URL for a directory instead of an HTML file, and when no default Web page is in the directory, Apache creates a page on the fly and displays it. This is usually only a listing of the different files in the directory. In effect, Apache indexes the items in the directory for you. You can set several options for generating and displaying such an index. If **FancyIndexing** is turned on, then Web page items are displayed with icons and column headers that can be used to sort the listing.

```
FancyIndexing on
```

Icon directives tells Apache what icon to display for a certain type of file. The **AddIconByType** and **AddIconByEncoding** directives use MIME-type information to determine the file's type and then associate the specified image with it. **AddIcon** uses the file's extension to determine its type. In the next example, the **text.gif** image is displayed for text files with the extension **.txt**. You can also use **AddIcon** to associate an image with a particular file. The **DefaultIcon** directive specifies the image used for files of undetermined type.

```
AddIcon /icons/text.gif .txt
DefaultIcon /icons/unknown.gif
AddIconByType (VID,/icons/movie.gif) video/*
```

With the **AddDescription** directive, you can add a short descriptive phrase to the filename entry. The description can be applied to an individual file or to filenames of a certain pattern.

```
AddDescription "Reunion pictures" /var/www/html/reunion.html
```

Within a directory, you can place special files that can be used to display certain text both before and after the generated listing. The **HeaderName** directive is used to set the name of the file whose text is inserted before the listing. The **ReadmeName** directive sets the name of the file whose text is placed at the end of the listing. You can use these directives in an **.htaccess** files or a **<Directory>** block to select particular files within a directory. The **ReadmeName** directive is usually set to README, and **HeaderName** to HEADER. In that case, Apache searches for files named **HEADER** and **README** in the directory.

```
HeaderName HEADER
ReadmeName README
```

With the **IndexOptions** directive, you can set different options for displaying a generated index. Options exist for setting the height and width of icons and filenames. The **IconsAreLinks** option makes icons part of filename anchors. The **ScanHTMLTitles** option reads the titles in HTML documents and uses those to display entries in the index listing instead of filenames. Various options exist for suppressing different index display features such as sorting, descriptions, and header/readme inserts. You can set options for individual directories using a Directory block or an **.htaccess** file. Normally, options set in higher-level directories are inherited by lower-level ones. If you use an **IndexOption** directive to set any new option, however, all previously inherited options are cleared. If you want to keep the inherited options, you can set options using the plus (+) or minus (**-**) symbols. These add or remove options. If you were also to set an option without the + or **-** symbols, though, all inherited options would be cleared.

```
IndexOptions IconsAreLinks FancyIndexing
IndexOptions +ScanHTMLTitles
```

Authentication

Your Web server can also control access on a per-user or per-group basis to particular directories on your Web site. You can require various levels for authentication. Access can be limited to particular users and require passwords, or expanded to allow members of a group access. You can dispense with passwords altogether or set up an anonymous type of access, as used with FTP.

To apply authentication directives to a certain directory, you place those directives within either a Directory block or the directory's **.htaccess** file. You use the **require** directive to determine what users can access the directory. You can list particular users or groups. The **AuthName** directive provides the authentication realm to the user, the name used to identify the particular set of resources accessed by this authentication process. The **AuthType** directive specifies the type of authentication, such as basic or digest. A **require** directive requires also **AuthType**, **AuthName**, and directives specifying the locations of group and user authentication files. In the following example, only the users george, robert, and mark are allowed access to the **newpics** directory:

```
<Directory /var/www/html/newpics
AuthType Basic
AuthName Newpics
AuthUserFile /web/users
AuthGroupFile /web/groups
<Limit GET POST>
 require users george robert mark
</Limit>
</Directory>
```

The next example allows group access by administrators to the CGI directory:

```
<Directory /var/www/html/cgi-bin
AuthType Basic
AuthName CGI
AuthGroupFile /web/groups
<Limit GET POST>
 require groups admin
</Limit>
</Directory>
```

To set up anonymous access for a directory, place the **Anonymous** directive with the user anonymous as its argument in the directory's Directory block or **.htaccess** file. You can also use the **Anonymous** directive to provide access to particular users without requiring passwords from them.

Apache maintains its own user and group authentication files specifying what users and groups are allowed to which directories. These files are normally simple flat files, such as your system's password and group files. They can become large, however, possibly slowing down authentication lookups. As an alternative, many sites have used database management files in place of these flat files. Database methods are then used to access the files, providing a faster response time. Apache has directives for specifying the authentication files, depending on the type of file you are using. The **AuthUserfile** and **AuthGroupFile** directives are used to specify the location of authentication files that have a standard flat file format. The **AuthDBUserFile** and **AuthDBGroupFile** directives are used for DB database files, and the **AuthDBMGUserFIle** and **AuthDBMGGroupFile** are used for DBMG database files.

The programs htdigest, htpasswd, and dbmmanage are tools provided with the Apache software package for creating and maintaining *user authentication files,* which are user password files listing users who have access to specific directories or resources on your Web site. htdigest and htpasswd manage a simple flat file of user authentication records, whereas dbmmanage uses a more complex database management format. If your user list is extensive, you may want to use a database file for fast lookups. htdigest takes as its arguments the authentication file, the realm, and the username, creating or updating the user entry. htpasswd can also employ encryption on the password. dbmmanage has an extensive set of options to add, delete, and update user entries. A variety of different database formats are used to set up such files. Three common ones are Berkeley DB2, NDBM, and GNU GBDM. dbmmanage looks for the system libraries for these formats in that order. Be careful to be consistent in using the same format for your authentication files.

Log Files

Apache maintains logs of all requests by users to your Web site. By default, these logs include records using the Common Log Format (CLF). The record for each request takes up a line composed of several fields: host, identity check, authenticated user (for logins), the date, the request line submitted by the client, the status sent to the client, and the size of the object sent in bytes. Using the **LogFormat** and **CustomLog** directives, you can customize your log record to add more fields with varying levels of detail. These directives use a format string consisting of field specifiers to determine the fields to record in a log record. You add whatever fields you want, and in any order. A field specifier consists of a percent (**%**) symbol followed by an identifying character. For example, **%h** is the field specifier for a remote host, **%b** for the size in bytes, and **%s** for the status. See the documentation for the mod_log_config module for a complete listing. Table 24-3 lists several of the commonly used ones. You should quote fields whose contents may take up more than one word. The quotes themselves must be quoted with a backslash to be included in the format string. The following example is the Common Log Format implemented as a **FormatLog** directive:

```
FormatLog "%h %l %u %t \"%r\" %s %b"
```

Instead of maintaining one large log file, you can create several log files using the **CustomLog** or **TransferLog** directive. This is helpful for virtual hosts where you may want to maintain a separate log file for each host. You use the **FormatLog** directive to define a default format for log records. **TransferLog** then uses this default as its format when creating a new log file. **CustomLog** combines both operations, enabling you to create a new file and to define a format for it.

```
FormatLog "%h %l %u %t \"%r\" %s %b"
# Create a new log file called myprojlog using the FormatLog format
TransferLog myprojlog
# Create a new log file called mypicslog using its own format
CustomLog mypicslog "%h %l %u %t \"%r\" %s %b"
```

Certain field specifiers in the log format can be qualified to record specific information. The **%i** specifier records header lines in requests the server receives. The reference for the specific header line to record is placed within braces between the **%** and the field specifier. For example, **User-agent** is the header line that indicates the browser software used in the request. To record User-agent header information, use the conversion specifier **%{User-agent}i**.

To maintain compatibility with NCSA servers, Apache originally implemented **AgentLog** and **RefererLog** directives to record User-agent and Referer headers. These have since been replaced by qualified **%i** field specifiers used for the **LogFormat** and **CustomLog** directives. A Referer header records link information from clients, detecting who may have links to your site. The following is an NCSA-compliant log format:

```
"%h %l %u %t \"%r\" %s %b\"%{Referer}i\" \"%{User-agent}i\"".
```

Apache provides two utilities for processing and managing log files. logresolve resolves IP addresses in your log file to host names. rotatelogs rotates log files without having to kill the server. You can specify the rotation time.

Virtual Hosting on Apache

Virtual hosting allows the Apache Web server to host multiple Web sites as part of its own. In effect, the server can act as several servers, each hosted Web site appearing separate to outside users. Apache supports both IP address and name-based virtual hosting. IP address virtual hosts use valid registered IP addresses, whereas name-based virtual hosts use fully qualified domain addresses. These domain addresses are provided by the host header from the requesting browser. The server can then determine the correct virtual host to use on the basis of the domain name alone. Note that SSL servers require IP virtual hosting. See **httpd.apache.org** for more information.

IP Address Virtual Hosts

In the IP address virtual hosting method, your server must have a different IP address for each virtual host. The IP address you use is already set up to reference your system. Network system administration operations can set up your machine to support several IP addresses. Your machine could have separate physical network connections for each one, or a particular connection could be configured to listen for several IP addresses at once. In effect, any of the IP addresses can access your system.

You can configure Apache to run a separate daemon for each virtual host, separately listening for each IP address, or you can have a single daemon running that listens for requests for all the virtual hosts. To set up a single daemon to manage all virtual hosts, use **VirtualHost** directives. To set up a separate daemon for each host, also use the **Listen** directive.

A **VirtualHost** directive block must be set up for each virtual host. Within each **VirtualHost** block, you place the appropriate directives for accessing a host. You should have **ServerAdmin**, **ServerName**, **DocumentRoot**, and **TransferLog** directives specifying the particular values for that host. You can use any directive within a **VirtualHost** block, except for **ServerType (1.3)**, **StartServers**, **MaxSpareServers**, **MinSpareServers**, **MaxRequestsPerChild**, **Listen**, **PidFile**, **TypesConfig**, **ServerRoot**, and **NameVirtualHost**.

Although you can use domain names for the address in the **VirtualHost** directive, using the actual IP address is preferable. This way, you are not dependent on your domain name service to make the correct domain name associations. Be sure to leave an IP address for your main server. If you use all the available IP addresses for your machine for virtual hosts, you can no longer access your main server. You could, of course, reconfigure your main server as a virtual host. The following example shows two IP-based virtual hosts blocks: one using an IP address, and the other a domain name that associates with an IP address:

```
<VirtualHost 192.168.1.23>
 ServerAdmin webmaster@mail.mypics.com
 DocumentRoot /groups/mypics/html
 ServerName www.mypics.com
 ErrorLog /groups/mypics/logs/error_log
 ....
</VirtualHost>

<VirtualHost www.myproj.org>
 ServerAdmin webmaster@mail.myproj.org
 DocumentRoot /groups/myproj/html
 ServerName www.myproj.org
 ErrorLog /groups/myproj/logs/error_log
 ....
</VirtualHost>
```

Name-Based Virtual Hosts

With IP-based virtual hosting, you are limited to the number of IP addresses your system supports. With name-based virtual hosting, you can support any number of virtual hosts using no additional IP addresses. With only a single IP address for your machine, you can still support an unlimited number of virtual hosts. Such a capability is made possible by the HTTP/1.1 protocol, which lets a server identify the name by which it is being accessed. This method requires the client, the remote user, to use a browser that supports the HTTP/1.1 protocol, as current browsers do (though older ones may not). A browser using such a protocol can send a host header specifying the particular host to use on a machine.

To implement name-based virtual hosting, use a **VirtualHost** directive for each host and a **NameVirtualHost** directive to specify the IP address you want to use for the virtual hosts. If your system has only one IP address, you need to use that address. Within the **VirtualHost** directives, you use the **ServerName** directive to specify the domain name you want to use for that host. Using **ServerName** to specify the domain name is important to avoid a DNS lookup. A DNS lookup failure disables the virtual host. The **VirtualHost** directives each take the same IP address specified in the **NameVirtualHost** directive as their argument. You use Apache directives within the **VirtualHost** blocks to configure each host separately. Name-based virtual hosting uses the domain name address specified in a host header to determine the virtual host to use. If no such information exists, the first host is used as the default. The following example implements two name-based virtual hosts. Here, **www.mypics.com** and **www.myproj.org** are implemented as name-based virtual hosts instead of IP-based hosts:

```
ServerName turtle.mytrek.com

NameVirtualHost 192.168.1.5

<VirtualHost 192.168.1.5>
 ServerName www.mypics.com
 ServerAdmin webmaster@mail.mypics.com
 DocumentRoot /var/www/mypics/html
 ErrorLog /var/www/mypics/logs/error_log
 ...
</VirtualHost>

<VirtualHost 192.168.1.5>
 ServerName www.myproj.org
 ServerAdmin webmaster@mail.myproj.org
 DocumentRoot /var/www/myproj/html
 ErrorLog /var/www/myproj/logs/error_log
 ...
</VirtualHost>
```

If your system has only one IP address, implementing virtual hosts prevents access to your main server with that address. You could no longer use your main server as a Web server directly; you could only use it indirectly to manage your virtual host. You could configure a virtual host to manage your main server's Web pages. You would then use your main server to support a set of virtual hosts that would function as Web sites, rather than the main server operating as one site directly. If your machine has two or more IP addresses, you can use one for the main server and the other for your virtual hosts. You can even mix IP-based virtual hosts and name-based virtual hosts on your server. You can also use separate IP addresses to support different sets of virtual hosts. You can further have several domain addresses access the same virtual host. To do so, place a **ServerAlias** directive listing the domain names within the selected **VirtualHost** block.

```
ServerAlias www.mypics.com www.greatpics.com
```

Requests sent to the IP address used for your virtual hosts have to match one of the configured virtual domain names. To catch requests that do not match one of these virtual hosts, you can set up a default virtual host using *_default_*:*. Unmatched requests are then handled by this virtual host.

```
<VirtualHost _default_:*>
```

Dynamic Virtual Hosting

If you have implemented many virtual hosts on your server that have basically the same configuration, you can use a technique called *dynamic virtual hosting* to have these virtual hosts generated dynamically. The code for implementing your virtual hosts becomes much smaller, and as a result, your server accesses them faster. Adding yet more virtual hosts becomes a simple matter of creating appropriate directories and adding entries for them in the DNS server.

To make dynamic virtual hosting work, the server uses commands in the mod_vhost_alias module (supported in Apache version 1.3.6 and up) to rewrite both the server name and the document root to those of the appropriate virtual server (for older Apache versions before 1.3.6, you use the mod_rewrite module). Dynamic virtual hosting can be either name-based or IP-based. In either case, you have to set the **UseCanonicalName** directive in such a way as to allow the server to use the virtual host name instead of the server's own name. For name-based hosting, you simply turn off **UseCanonicalName**. This allows your server to obtain the hostname from the host header of the user request. For IP-based hosting, you set the **UseCanonicalName** directive to DNS. This allows the server to look up the host in the DNS server.

```
UseCanonicalName Off
UseCanonicalName DNS
```

You then have to enable the server to locate the different document root directories and CGI bin directories for your various virtual hosts. You use the **VirtualDocumentRoot** directive to specify the template for virtual hosts' directories. For example, if you place the different host directories in the **/var/www/hosts** directory, then you could set the **VirtualDocumentRoot** directive accordingly.

```
VirtualDocumentRoot /var/www/hosts/%0/html
```

The **%0** will be replaced with the virtual host's name when that virtual host is accessed. It is important that you create the dynamic virtual host's directory using that host's name. For example, for a dynamic virtual host called **www.mygolf.org**, you would create a directory named **/var/www/hosts/www.mygolf.org**. Then create subdirectories for the document root and CGI programs as in **/var/www/hosts/www.mygolf.org/html**. For the CGI directory, use the **VirtualScriptAlias** directive to specify the CGI subdirectory you use.

```
VirtualScriptAlias /var/www/hosts/%0/cgi-bin
```

A simple example of name-based dynamic virtual hosting directives follows:

```
UseCanonicalName Off
VirtualDocumentRoot /var/www/hosts/%0/html
VirtualScriptAlias /var/www/hosts/%0/cgi-bin
```

If a request was made for **www.mygolf.com/html/mypage**, that would evaluate to

```
/var/www/hosts/www.mygolf.com/html/mypage
```

The mod_vhots_alias module supports various interpolated strings, each beginning with a **%** symbol and followed by a number. As you have seen, **%0** references the entire Web address. **%1** references only the first segment, **%2** references the second, **%-1** references the last part, and **%2+** references from the second part on. For example, if you only want to use the second part of a Web address for the directory name, you would use the following directives:

```
VirtualDocumentRoot /var/www/hosts/%2/html
VirtualScriptAlias /var/www/hosts/%2/cgi-bin
```

In this case, a request made for **www.mygolf.com/html/mypage** would use only the second part of the Web address. This would be "mygolf" in **www.mygolf.com**, and would evaluate to

```
/var/www/hosts/mygolf/html/mypage
```

If you used **%2+** instead, as in **/var/www/hosts/%2/html**, then the request for **www.mygolf.com/html/mypage** would evaluate to

```
/var/www/hosts/mygolf.com/html/mypage
```

The same method works for IP addresses, where **%1** references the first IP address segment, **%2** references the second, and so on.

A simple example of dynamic virtual hosting is shown here:

```
UseCanonicalName Off

NameVirtualHost 192.168.1.5

<VirtualHost 192.168.1.5>
 ServerName www.mygolf.com
 ServerAdmin webmaster@mail.mygolf.com
 VirtualDocumentRoot /var/www/hosts/%0/html
 VirtualScriptAlias /var/www/hosts/%0/cgi-bin
 ...
</VirtualHost>
```

To implement IP-based dynamic virtual hosting instead, set the **UseCanonicalName** to DNS instead of Off.

```
UseCanonicalName DNS
VirtualDocumentRoot /var/www/hosts/%0/html
VirtualScriptAlias /var/www/hosts/%0/cgi-bin
```

One drawback of dynamic virtual hosting is that you can only set up one log for all your hosts. However, you can create your own shell program to simply cut out the entries for the different hosts in that log.

```
LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon
```

Implementing IP-based dynamic virtual hosting in the standard way as shown previously will slow down the process, as your server will have to perform a DNS lookup to discover the name of your server using its IP address. You can avoid this step by simply using the IP address for your virtual host's directory. So, for IP virtual host 192.198.1.6, you would create a directory **/var/www/hosts/192.198.1.6**, with an **html** subdirectory for that host's document root. You would use the **VirtualDocumentRootIP** and **VirtualScriptAliasIP** directives to use IP addresses as directory names. Now the IP address can be mapped directly to the document root directory name, no longer requiring a DNS lookup. Also be sure to include the IP address in your log, **%A**.

```
UseCanonicalName DNS
LogFormat "%A %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon

VirtualDocumentRootIP /var/www/hosts/%0/html
VirtualScriptAliasIP /var/www/hosts/%0/cgi-bin
```

You can mix these commands in with other virtual host entries as you need them. For example, to specify the document root directory for a nondynamic name-based virtual host, you could simply use the **VirtualDocumentRoot** directive. In other words, you can simply use the same directories for both dynamic and nondynamic virtual hosts. You could still specify other directories for different nondynamic virtual hosts as you wish. In the following example, the **www.mypics.com** name-based virtual host uses the dynamic virtual host directive **VirtualDocumentRoot** to set its document root directory. It now uses **/var/www/www.mypics.com/html** as its document root directory. The CGI directory, however, is set as a nondynamic directory, **/var/www/mypics/cgi-bin**.

```
UseCanonicalName Off

NameVirtualHost 192.168.1.5

<VirtualHost 192.168.1.5>
 ServerName www.mypics.com
 ServerAdmin webmaster@mail.mypics.com
 VirtualDocumentRoot /var/www/%0/html
 ScriptAlias /var/www/mypics/cgi-bin
 ...
</VirtualHost>
```

Server-Side Includes

Server-side includes (SSIs) are designed to provide a much more refined control of your Web site content, namely the Web pages themselves. Server-side includes are Apache directives placed within particular Web pages as part of the page's HTML code. You can configure your Apache Web server to look for SSI directives in particular Web pages and execute them. First, you have to use the **Options** directive with the **includes** option to allow SSI directives.

```
Options Includes
```

You need to instruct the server to parse particular Web pages. The easiest way to enable parsing is to instruct Apache to parse HTML files with specified extensions. Usually, the extension **.shtml** is used for Web pages that have SSI directories. In fact, in the default Apache configuration files, you can find the following entry to enable parsing for SSI directives in HTML files. The **AddType** directive here adds the **.shtml** type as an HTML type of file, and the **AddHandler** directive specifies that **.shtml** files are to be parsed (server-parsed).

```
# To use server-parsed HTML files
AddType text/html .shtml
AddHandler server-parsed .shtml
```

Instead of creating a separate type of file, you can use the **XBitHack** directive to have Apache parse any executable file for SSI directives. In other words, any file with execute permission (see Chapter 9) will be parsed for SSI directives.

SSI directives operate much like statements in a programming language. You can define variables, create loops, and use tests to select alternate directives. An SSI directive consists of an element followed by attributes that can be assigned values. The syntax for a SSI directive is shown here:

```
<!--#element attribute=value … -->
```

You can think of an element as operating much like a command in a programming language and attributes as its arguments. For example, to assign a value to a variable, you use the **set** element with the variable assignment as its attribute. The **if** directive displays any following text on the given Web page. The **if** directive takes as its attribute **expr**, which is assigned the expression to test. The test is able to compare two strings using standard comparison operators like **<=**, **!=**, or **=**. Variables used in the test are evaluated with the **$** operator.

```
<!--#set myvar="Goodbye" -->
<!--#if expr="$myvar = Hello" -->
```

Other helpful SSI elements are **exec**, which executes CGI programs, or shell commands, which read the contents of a file into the Web page and also execute CGI files. The **echo** element displays values such as the date, the document's name, and the page's URL. With the **config** element, you can configure certain values, such as the date or file size.

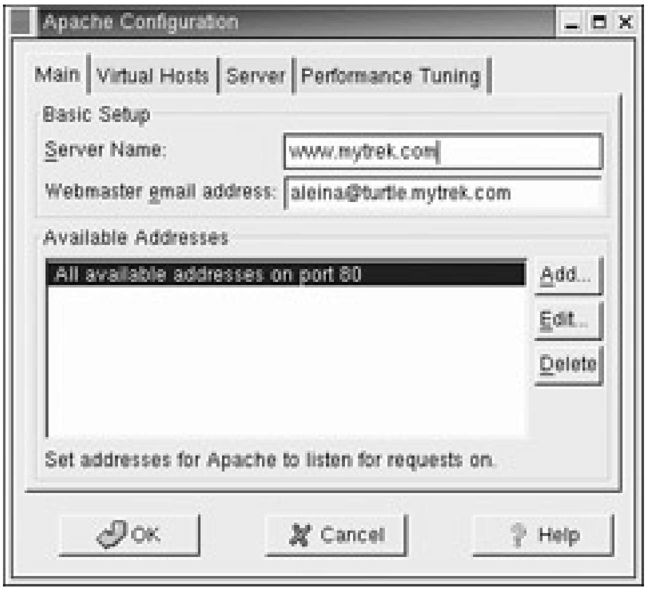
Apache GUI Configuration Tools

Red Hat 7.1 provides a GUI configuration tool called the Apache Configuration Tool, accessible from the Gnome and KDE desktops. Also available is Comanche, a popular Apache configuration tool that you download from the Internet.

Note The Apache GUI Project (**gui.apache.org**) provides a set of GUI tools for configuring and managing your Apache Web server. Its currently active projects are Comanche and TkApache. In the Linuxconf utility, you can also configure your Apache Web server. Webmin provides a very complete Apache Web server module.

Apache Configuration Tool

The Apache Configuration Tool opens with a window displaying panels for Main, Virtual Hosts, Server, and Performance Tuning. In each of these you will see buttons to open up dialog boxes where you can enter default settings. You will also be able to enter settings for particular items like virtual hosts and particular directories. For example, in the Virtual Hosts panel you can enter in default settings for all virtual hosts, as well as add and edit particular virtual hosts. Click the Help button to display a Web page-based reference manual that details how to use each panel. On the Main panel, you enter your Web server address, the Webmaster's e-mail address, and the ports the Web server will be listening on (see Figure 24-1).



Server Name: www.mytrek.com

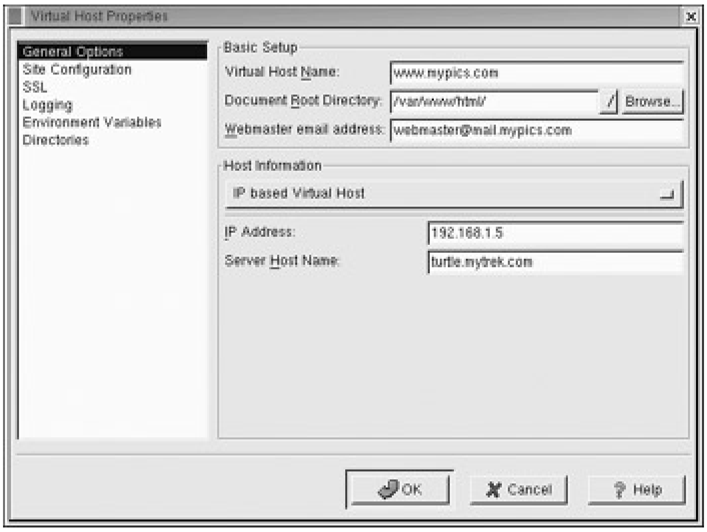
Webmaster email address: aleina@turtle.mytrek.com

All available addresses on port 80

Set addresses for Apache to listen for requests on.

Figure 24-1: Apache Configuration Tool

On the Virtual Hosts panel, be sure to click the Edit Default Settings button to set the default settings for pages searches, error codes, log files, and directories. To add a virtual host, click on the Add button to open a window where you can enter host information such as the virtual hostname and IP address (see Figure 24-2). On the sidebar, you can select different configuration panels for the virtual host, such as log files and directory controls.



Virtual Host Name: www.mypics.com

Document Root Directory: /var/www/html/

Webmaster email address: webmaster@mail.mypics.com

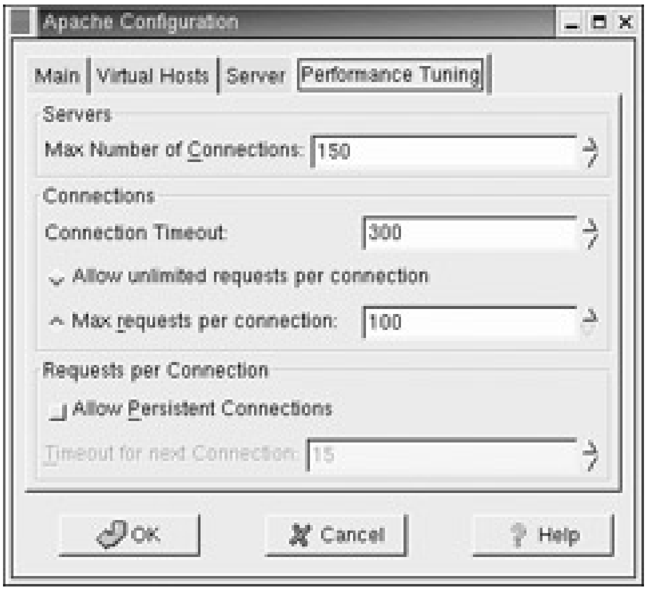
IP based Virtual Host

IP Address: 192.168.1.5

Server Host Name: turtle.mytrek.com

Figure 24-2: Virtual Host configuration

On the server panel, you set administrative settings such as the Apache server's user ID and the process ID file. The Performance Tuning panel lets you set different usage limits such as the maximum number of requests and the number of requests per connection (see Figure 24-3).


Apache Configuration

Servers

Max Number of Connections: 150

Connections

Connection Timeout: 300

○ Allow unlimited requests per connection

● Max requests per connection: 100

Requests per Connection

☐ Allow Persistent Connections

Timeout for next Connection: 15

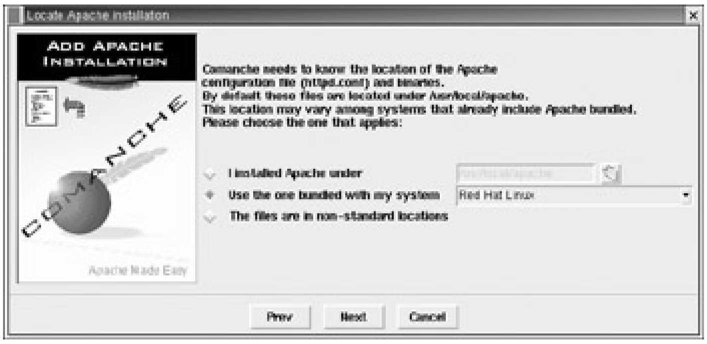
Figure 24-3: Performance Tuning panel

When the Apache Configuration Tool saves its settings, it will overwrite the Apache configuration file, **/etc/httpd/conf/httpd.conf**. It is advisable that you first make a backup copy of your **httpd.conf** file in case you want to restore the original settings created by Red Hat for Apache. If you have already manually edited this file, you will receive a warning, and the Apache Configuration Tool will make a backup copy in **/etc/httpd/conf/httpd.conf.bak.**

Comanche

Comanche (Configuration Manager for Apache) is an easy-to-use, full-featured Apache configuration utility that runs on any X Windows window manager. You can download the current version and documentation from the Comanche Web site at **www.comanche.org**. Comanche uses a simple, directory tree-like structure to enable you to access and configure your main Web server, and any virtual server you have set up. Currently, Comanche can only configure the server on your local machine, but in the future it may be able to configure remote servers.

When you first start the program, a window is displayed where you select the kind of Apache installation you have. For Red Hat, select the Red Hat Linux in the "Use the one bundled with my system" entry (see Figure 24-4).


Locate Apache installation

Comanche needs to know the location of the Apache configuration file (httpd.conf) and binaries.
By default these files are located under /usr/local/apache.
This location may vary among systems that already include Apache bundled.
Please choose the one that applies:

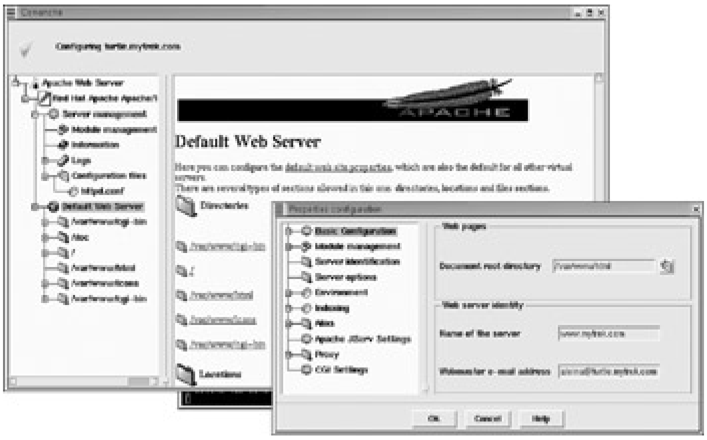
○ I installed Apache under /usr/local/apache

● Use the one bundled with my system Red Hat Linux

○ The files are in non-standard locations

Figure 24-4: Comanche server information

The main Comanche window has a sidebar that shows a tree of Apache servers, and the main panel shows the items in the selected entry in the tree (see Figure 24-5). The main tree entry will display the Apache Web server and under it, the Red Hat Apache server. Here you will find entries for Server Management and the Default Web server, the primary Web server. To perform actions on any of these entries, select the entry and then select the list of actions on the main panel. The Server Management entry displays a list of items to start, stop, restart, and query the server status, as well as to save your configuration. Any changes you make with Comanche are not made in your Apache configuration files until you explicitly save the configuration. Be sure to save before you quit. When you save your configuration, this version overwrites your **httpd.conf** file.



Here you can configure the default web site properties, which are also the default for all other virtual servers.
There are several types of sections allowed in this one: directories, locations and files sections.

Document root directory /var/www/html

Name of the server www.mytrek.com

Webmaster e-mail address [illegible]@turtle.mytrek.com

Figure 24-5: The Comanche Apache configuration tool

To configure your Default Web server, click on it and then select the default configuration entry on the mail panel. This opens a dialog box like that shown in Figure 24-5. If you double-click the Default Web server icon, the tree expands to list server locations. These are the root directory, your Web server home directory, the CGI program directory, and the documentation directory. To create new directories, right-click the Web server entry and select the New item in the menu that pops up.

Initially, only one server is listed, labeled Default Web Server. If you add virtual hosts, they are listed at the same level. To create a virtual host, right-click the Apache Server entry and then select New on the pop-up menu. This activates a drop-down menu with a Virtual Host entry. Select the Virtual Host entry to create your virtual host.

To configure a directory, right-click its entry and select the Properties item from the pop-up menu displayed. This opens a new window for configuring your selection, similar to the one shown in Figure 24-5. The window is divided into sidebar and main panel: The sidebar shows a tree of configuration items and the main panel displays the configuration dialog windows for these items. For example, by selecting the Basic configuration entry, you can display text boxes for entering items such as the document root directory and the name of your server. Other entries let you set server options, proxy settings, and indexing formats.

Web Server Security-SSL

Web server security deals with two different tasks: protecting your Web server from unauthorized access, and providing security for transactions carried out between a Web browser client and your Web server. To protect your server from unauthorized access, you use a proxy server such as Squid. Squid is a GNU proxy server often used with Apache on Linux systems. (See Chapter 28 for a detailed explanation of the Squid server.) Apache itself has several modules that provide security capabilities. These include mod_access for mandatory controls; mod_auth, mod_auth_db, mod_auth_digest, and mod_auth_dbm, which provide authentication support; and mod_auth_anon for anonymous FTP-like logging (see previous sections on access control and authentication).

To secure transmissions, you need to perform three tasks. You have to verify identities, check the integrity of the data, and insure the privacy of the transmission. To verify the identities of the hosts participating in the transmission, you perform authentication procedures. To check the integrity of the data you add digital signatures containing a digest value for the data. The digest value is a value that uniquely represents the data. Finally, to secure the privacy of the transmission, you encrypt it. Transactions between a browser and your server can then be encrypted, with the browser and your server alone able to decrypt the transmissions. The protocol most often used to implement secure transmissions with Linux Apache Web servers is the Secure Sockets Layer (SSL) protocol, which was originally developed by Netscape for secure transactions on the Web.

Like the Secure Shell (SSH) described in Chapter 40 and GPG discussed in Chapter 6, SSL uses a form of public- and private-key encryption for authentication. Data is encrypted with the public key but can only be decrypted with the private key. Once authenticated, an agreed-upon cipher is used to encrypt the data. Digital signatures encrypt an MD5 digest value for data to ensure integrity. Authentication is carried out with the use of certificate authority. Certificates identify the different parties in a secure transmission, verifying that they are who they say they are. A Web server will have a certificate verifying its identity, verifying that it is the server it claims to be. The browser contacting the server will also have a certificate identifying who it is. These certificates are, in turn, both signed by a certificate authority, verifying that they are valid certificates. A certificate authority is an independent entity that both parties trust.

A certificate contains the public key of the particular server or browser it is given to, along with the digital signature of the certificate authority and identity information such as the name of the user or company running the server or browser. The effectiveness of a certificate depends directly on the reliability of the certificate authority issuing it. To run a secure Web server on the Internet, you should obtain a certificate from a noted certificate authority such as Verisign. A commercial vendor such as Stronghold can do this for you. Many established companies may already maintain their own certificate authority, securing transmissions within their company networks. An SSL session is set up using a handshake sequence in which the server and browser are authenticated by exchanging certificates, a cipher is agreed upon to encrypt the transmissions, and the kind of digest integrity check is chosen. There is also a choice in the kind of public key encryption used for authentication, either RSA or DSA. For each session a unique session key is set up that the browser and server use.

A free version of SSL called OpenSSL is available for use with Apache (see **www.openssl.org**). It is based on SSLeay from Eric A. Young and Tim J. Hudson. However, U.S. government restrictions prevent the Apache Web server from being freely distributed with SSL capabilities built-in. You have to separately obtain SSL and update your Apache server to incorporate this capability.

The U.S. government maintains export restrictions on encryption technology over 40 bits. SSL, however, supports a number of ciphers using 168-, 128-, and 40-bit keys (128 is considered secure, and so by comparison the exportable 40-bit versions are useless). This means that if Apache included SSL, it could not be distributed outside the U.S. Outside the U.S., however, there are projects that do distribute SSL for Apache using OpenSSL. These are free for noncommercial use in the U.S., though export restrictions apply. The Apache-SSL project freely distributes Apache with SSL built-in, apache+ssl. You can download this from their Web site at **www.apache-ssl.org** (though there are restrictions on exporting encryption technology, there is none on importing it). In addition, the mod_ssl project provides an SSL module with patches that you can use to update your Apache Web server to incorporate SSL (**www.modsll.org**). mod_ssl is free for both commercial and noncommercial use under an Apache style license.

Red Hat includes the mod_ssl module with its distribution in the mod_ssl package.

```
rpm -i mod_ssl-2.8.1-5.i386.rpm
```

The mod_ssl implementation of SSL provides an alternate access to your Web server using a different port (443) and a different protocol, https. In effect, you have both an SSL server and a nonsecure version. To access the secure SSL version you use the protocol https instead of http for the Web server's URL address. For example, to access the SSL version for the Web server running at **www.mytrek.com**, you would use the protocol https in its URL, as shown here.

```
https://www.mytrek.com
```

You can configure mod_ssl using a number of configuration directives in the Apache configuration file, smb.conf. On Red Hat, the default configuration file installed with Apache contains a section for the SSL directives along with detailed comments. Check the online documentation for mod_ssl at **www.modssl.org** for a detailed reference listing all the directives. There are global, server-based, and directory-based directives available.

In the Red Hat smb.conf file, the inclusion of SSL directives is controlled by IfDefine blocks enabled by the HAVE_SSL flag. For example, the following code will load the SSL module.

```
<IfDefine HAVE_SSL>
LoadModule ssl_module        modules/libssl.so
</IfDefine>
```

The SSL version for your Apache Web server is set up in the smb.conf file as a virtual host. The SSL directives are enabled by an ifDefine block using the HAVE_SSL flag. Several default directives are implemented such as the location of SSL key directories and the port that the SSL version of the server will listen on (443). Others are commented out. You can enable them by removing the preceding # symbol, setting your own options. Several of the directives are shown here.

```
<IfDefine HAVE_SSL>
## SSL Virtual Host Context

#  Apache will only listen on port 80 by default.  Defining the
 virtual server
#  (below) won't make it automatically listen on the virtual server's port.
```

```
Listen 443

<VirtualHost _default_:443>

#  General setup for the virtual host
DocumentRoot "/var/www/html"

#   SSL Engine Switch:
#   Enable/Disable SSL for this virtual host.
SSLEngine on
#SSLCipherSuite
ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

#   Server Certificate:
SSLCertificateFile /etc/httpd/conf/ssl.crt/server.crt
#SSLCertificateFile /etc/httpd/conf/ssl.crt/server-dsa.crt

#   Server Private Key:
SSLCertificateKeyFile /etc/httpd/conf/ssl.key/server.key

#   Certificate Authority (CA):
#SSLCACertificatePath /etc/httpd/conf/ssl.crt
#SSLCACertificateFile /etc/httpd/conf/ssl.crt/ca-bundle.crt
```

In the **/etc/httpd/conf** directory, mod_ssl will set up several SSL directories that will contain SSL authentication and encryption keys and data. The ssl.crt directory will hold certificates for the server. The ssl.key directory holds the public and private keys used in authentication encryption. Revocation lists for revoking expired certificates are kept in ssl.crl. The ssl.csr directory holds the certificate signing request used to request an official certificate from a certificate authority. ssl.prm holds parameter files used by the DSA key encryption method. Check the README files in each directory for details on the SSL files they contain.

The mod_ssl installation will provide you with a demonstration certificate called snakeoil that you can use to test your SSL configuration. When you have an official certificate, you can install it with the **make certificate** command within the ssl.crt directory. This will overwrite the server.crt server certificate file.

Chapter 25: Domain Name Service

Overview

The Domain Name Service (DNS) is an Internet service that converts domain names into their corresponding IP addresses. As you may recall, all computers connected to the Internet are addressed using an Internet Protocol (IP) address. The IP address may be implemented in either the newer IPV6 (Internet Protocol Version 6) format or on the older and more common IPV4 (Internet Protocol Version 4) format. Since most systems still use the IPV4 addressing, that format will be used in these examples. In the older IPV4 format, the IP address consists of a number composed of four segments separated by periods. Depending on the type of network, several of the first segments are used for the network address and several of the last segments are used for the host address. In a standard class C network used in smaller networks, the first three segments are the computer's network address, and the last segment is the computer's host ID (as used in these examples). For example, in the address 192.168.1.2, 192.168.1 is the network address and 2 is the computer's host ID within that network.

Together, they make up an IP address with which the computer can be addressed from anywhere on the Internet. IP addresses, though, are difficult to remember and easy to get wrong.

As a normal user on a network might have to access many different hosts, keeping track of the IP addresses needed quickly became a problem. It was much easier to label hosts with names and use the names to access them. Names were associated with IP addresses. When a user used a name to access a host, the corresponding IP address was looked up first and then used to provide access.

IP addresses were associated with corresponding names, called fully qualified domain names. A *fully qualified domain name* is composed of three or more segments: The first segment is the name to identify the host, and the remaining segments are for the network in which the host is located. The network segments of a fully qualified domain name are usually referred to simply as the domain name, while the host part is referred to as the hostname (though this is also used to refer to the complete fully qualified domain name). In effect, subnets are referred to as domains. The fully qualified domain name, **www.linux.org**, has an IP address 198.182.196.56, where 198.182.196 is the network address and 56 is the host ID. Computers can be accessed only with an IP address. So, a fully qualified domain name must first be translated into its corresponding IP address to be of any use. The parts of the IP address that make up the domain name and the hosts can vary. See Chapter 38 for a detailed discussion of IP addresses, including network classes and Classless Interdomain Routing (CIDR).

Any computer on the Internet can maintain a file that manually associates IP addresses with domain names. On Linux and Unix systems, this file is called the **/etc/hosts** file. Here, you can enter the IP addresses and domain names of computers you commonly access. Using this method, however, each computer needs a complete listing of all other computers on the Internet, and that listing must be updated constantly. Early on, this became clearly impractical for the Internet, though it is still feasible for small isolated networks. The Domain Name Service has been implemented to deal with the task of translating the domain name of any computer on the Internet to its IP address. The task is carried out by interconnecting servers that manage the Domain Name Service (also referred to either as DNS servers or as name servers). These DNS servers keep lists of fully qualified domain names and their IP addresses, matching one up with the other. This service that they provide to a network is referred to as the Domain Name Service. The Internet is composed of many connected subnets called *domains,* each with its own Domain Name Service (DNS) servers that keep track of all the fully qualified domain names and IP addresses for all the computers on its network. DNS servers are hierarchically linked to root servers, which, in turn, connect to other root servers and the DNS servers on their subnets throughout the Internet. The section of a network for which a given DNS server is responsible is called a *zone.* Although a zone may correspond to a domain, many zones may, in fact, be within a domain, each with its own name server. This is true for large domains where too many systems exist for one name server to manage.

When a user enters a fully qualified domain name to access a remote host, a resolver program queries the local network's DNS server requesting the corresponding IP address for that remote host. With the IP address the user can then access the remote host. In Figure 25-1, the user at **rabbit.mytrek.com** wants to connect to the remote host **lizard. mytrek.com**. **rabbit.mytrek.com** first sends a request to the network's DNS server-in this case, **turtle.mytrek.com**-to look up the name **lizard.mytrek.com** and find its IP address. It then returns the IP address for **lizard.mytrek.com**, 192.168.1.3, to the requesting host,

rabbit.mytrek.com. With the IP address, the user at **rabbit.mytrek.com** can then connect to **lizard.mytrek.com**.

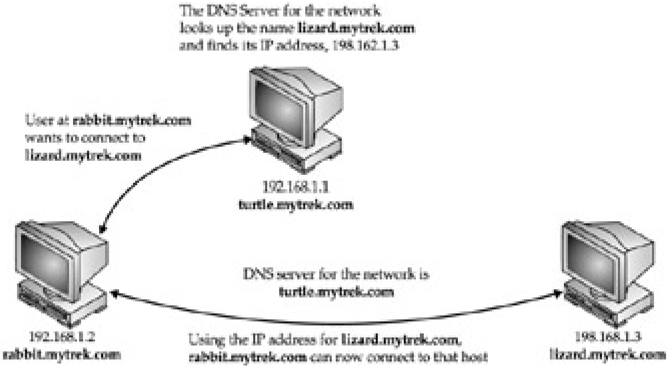

The DNS Server for the network looks up the name lizard.mytrek.com and finds its IP address, 198.162.1.3
User at rabbit.mytrek.com wants to connect to lizard.mytrek.com
192.168.1.1
turtle.mytrek.com
DNS server for the network is turtle.mytrek.com
192.168.1.2
rabbit.mytrek.com
Using the IP address for lizard.mytrek.com, rabbit.mytrek.com can now connect to that host
198.168.1.3
lizard.mytrek.com

Figure 25-1: DNS server operation

The names of the DNS servers that service a host's network are kept in the host's **/etc/resolv.conf** file. When setting up an Internet connection, the name servers provided by your Internet service provider (ISP) were placed in this file. These name servers resolve any fully qualified domain names that you use when you access different Internet sites. For example, when you enter a Web site name in your browser, the name is looked up by the name servers, and the name's associated IP address is then used to access the site.

Local Area Network Addressing

If you are setting up a DNS server for a local area network (LAN) that is not connected to the Internet, you should use a special set of IP numbers reserved for such non-Internet networks (also known as *private networks* or *intranets*). This is especially true if you are implementing IP masquerading, where only a gateway machine has an Internet address, and the others make use of that one address to connect to the Internet. For a class C network (254 hosts or less), these are numbers that have the special network number 192.168, as used in these examples. If you are setting up a LAN, such as a small business or home network, you are free to use these numbers for your local machines. You can set up a private network, such as an intranet, using network cards such as Ethernet cards and Ethernet hubs, and then configure your machines with IP addresses starting from 192.168.1.1. The host segment can range from 1 to 254, where 255 is used for the broadcast address. If you have three machines on your home network, you can give them the addresses 192.168.1.1, 192.168.1.2, and 192.168.1.3. You can then set up domain name services for your network by running a DNS server on one of the machines. This machine becomes your network's DNS server. You can then give your machines fully qualified domain names and configure your DNS server to translate the names to their corresponding IP addresses. As shown in Figure 25-2, for example, you could give the machine 192.168.1.1 the name **turtle.mytrek.com**, and the machine 192.168.1.2 the name **rabbit.mytrek.com**. You can also implement Internet services on your network such as FTP, Web, and mail services by setting up servers for them on your machines. You can then configure your DNS server to let users access those services using fully qualified domain names. For example, for the **mytrek.com** network, the Web server could be accessed using the name www.mytrek.com. Instead of a Domain Name Service, you could have the **/etc/hosts** files in each machine contain the entire list of IP addresses and domain names for all the machines in your network. But, for any changes, you would have to update each machine's **/etc/hosts** file.

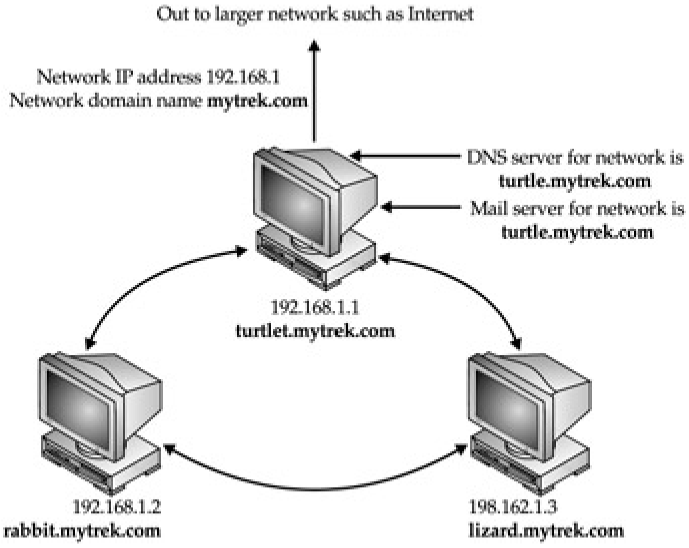

Out to larger network such as Internet
Network IP address 192.168.1
Network domain name **mytrek.com**
DNS server for network is
turtle.mytrek.com
Mail server for network is
turtle.mytrek.com
192.168.1.1
turtlet.mytrek.com
192.168.1.2
rabbit.mytrek.com
198.162.1.3
lizard.mytrek.com

Figure 25-2: DNS server and network

Numbers are also reserved for class A and class B non-Internet local networks. Table 25-1 lists these addresses. The possible addresses available span from 0 to 255 in the host segment of the address. For example, class B network addresses range from 172.16.0.0 to 172.16.255.255, giving you a total of 65,534 possible hosts. The class C network ranges from 192.168.0.0 to 192.168.255.255, giving you 254 possible subnetworks, each with 254 possible hosts. The number 127.0.0.0 is reserved for a system's loopback interface, which allows it to communicate with itself, as it enables users on the same system to send messages to each other.

These numbers were originally designed for class-based addressing. However, they can just as easily be used for Classless Interdomain Routing (CIDR) addressing, where you can create subnetworks with a smaller number of hosts. For example, the 254 hosts addressed in a class C network could be split into two subnetworks, each with 125 hosts. See Chapter 39 for more details.

Table 25-1: Non-Internet Private Network IP Addresses

Address	**Network**
10.0.0.0	Class A network
172.16.0.0 to 172.31.255.255	Class B network
192.168.0.0	Class C network
127.0.0.0	Loopback network (for system self-communication)

BIND

The DNS server software currently in use on Linux systems is Berkeley Internet Name Domain (BIND). BIND was originally developed at the University of California, Berkeley, and is currently maintained and supported by the Internet Software Consortium (ISC). You can obtain BIND information and current software releases from its Web site at **www.isc.org**. Web page documentation and manuals are included with the software package. RPM packages are available at the Red Hat FTP site. The BIND directory in **/usr/share/doc** contains extensive documentation, including Web page manuals and examples. The Linux

HOW-TO for the Domain Name Service, DNS-HOWTO, provides detailed examples. Documentation, news, and DNS tools can be obtained from the DNS Resource Directory (DNSRD) at **www.dns.net/dnsrd**. The site includes extensive links and online documentation, including the *BIND Operations Guide* (*BOG*). See Table 25-2 for a list of DNS resources.

Note Several alternative DNS servers are now available. These include djbdns, noted for its security features; CustomDNS, a dynamic server implemented in Java (**customdns.sourceforge.net**); and Yaku-NS, an embedded server. djbdns (**dgbdns.org**), written by D. J. Bernstein, is designed specifically with security in mind, providing a set of small server daemons, each performing specialized tasks. In particular, djbdns separates the name server, caching server, and zone transfer tasks into separate programs. tinydns implements the authoritative name server for a network, whereas dnscache implements a caching server that will resolve requests form DNS clients like Web browsers. In effect, dnscache operates as the name server that your applications will use to resolve addresses. dnscache will then query tinydns to resolve addresses on your local network. Zone transfers are handled separately by axfrdns and asfget.

Currently ISC has contracted with two companies, Nominum and Mind, to provide BIND support. Nominum is an ISC support partner and has taken an active role in BIND development. At its Web site at **www.nominum.com**, you can find BIND documentation, including the BIND 9 Administrator's Reference. Nominum, like many commercial companies that support open source software, provides professional consultant and support services, while freely contributing to open source development. Mind provides consulting services for the European market.

Table 25-2: BIND Resources

Web site	Resource
www.isc.org	Internet Software Consortium
www.dns.net/dnsrd	DNS Resource Directory
www.nominum.com	Nominum, BIND support and consulting
mind.be	Mind, BIND support and consulting for Europe

The BIND DNS server software consists of a name server daemon called **named**, several sample configuration files, and resolver libraries. As of 1998, a new version of BIND, beginning with the series number 8.x, implemented a new configuration file using a new syntax. Recently version 9.0 was released, adding new security features. Older versions, which begin with the number 4.x, use a different configuration file with an older syntax. Red Hat currently installs the newer 9.x version of BIND.

The name of the BIND name server daemon is **named**. To operate your machine as a name server, simply run the **named** daemon with the appropriate configuration. The **named** daemon listens for resolution requests and provides the correct IP address for the requested host name. You can use the Name Daemon Controller, rndc, utility provided with BIND to start, stop, restart, and check the status of the server as you test its configuration. rndc with the **stop** command stops **named** and, with the **start** command, starts it again, reading your **named.conf** file. rndc with the **help** command provides a list of all rndc commands. Once your name server is running, you can test it using the dig or nslookup utilities, which queries a name server, providing information about hosts and domains. If you start dig with no arguments, it enters an interactive mode where you can issue different dig commands to refine your queries. Numerous other DNS tools are also available, such as nslint and host. Check the DNS Resource Directory at **www.dns.net/dnsrd** for a listing. Table 25-3 lists several DNS administrative tools.

Table 25-3: BIND Diagnostic and Administration Tools

Tool	Description
dig *domain*	Domain Information Groper, tool to obtain information on a DNS server. Preferred over nslookup.
host *hostname*	Simple lookup of hosts.
nslookup *domain*	Tool to query DNS servers for information about domains and hosts.
rndc *command*	Remote Name Daemon Controller is an administrative tool for managing a DNS server (version 9.x).
ndc	Name Daemon Controller (version 8.x).

On Red Hat systems, the **named** daemon is started using a startup script in the **/etc/rc.d/init.d** directory called **named**. You can use this script to start, stop, and restart the daemon using the **stop**, **start**, and **restart** arguments. You can invoke the script with the **service** command, as shown here.

```
service named restart
```

On Red Hat systems, **named** runs as a standalone daemon, starting up when the system boots and constantly runs. If you don't want **named** to start up automatically, you can use the System V Runlevel Editor or Setup to change its status.

Domain Name Service Configuration

You configure a DNS server using a configuration file, several zone files, and a cache file. The part of a network for which the name server is responsible is called a zone. A *zone* is not the same as a domain, because in a large domain you could have several zones, each with its own name server. You could also have one name server service several zones. In this case, each zone has its own zone file. The zone files hold resource records that provide hostname and IP address associations for computers on the network for which the DNS server is responsible. Zone files exist for the server's network and the local machine. Zone entries are defined in the **named.conf** file. Here, you place zone entries for your master, slave, and forward DNS servers. The most commonly used zone types are described here:

- **Master zone** This is the primary zone file for a network. It holds the mappings from domain names to IP addresses for all the hosts on the network.
- **Slave zone** These are references to other DNS servers for your network. Your network can have a master DNS server and several slave DNS servers to help carry the workload. A slave DNS server automatically copies its configuration files, including all zone files, from the master DNS server. Any changes to the master configuration files trigger an automatic download of these files to the slave servers. In effect, you

only have to manage the configuration files for the master DNS server, as they are automatically copied to the slave servers.

- **Forward zone** The forward zone lists name servers outside your network that should be searched if your network's name server fails to resolve an address.
- **IN-ADDR.ARPA zone** DNS can also provide reverse resolutions, where an IP address is used to determine the associated domain name address. Such lookups are provided by **IN-ADDR.ARPA** zone files. Each master zone file usually has a corresponding **IN-ADDR.ARPA** zone file to provide reverse resolution for that zone. For each master zone entry, a corresponding reverse mapping zone entry named **IN-ADDR.ARPA** also exists, as well as one for the localhost. This entry performs reverse mapping from an IP address to its domain name. The name of the zone entry uses the domain IP address, which is the IP address with segments listed starting from the host, instead of the network. So, for the IP address 192.168.1.4 where 4 is the host address, the corresponding domain IP address is 4.1.168.192, listing the segments in reverse order. The reverse mapping for the localhost is 0.0.127.
- **Hint zone** A hint zone specifies the root name servers and is denoted by a period (.).A DNS server is normally connected to a larger network, such as the Internet, which has its own DNS servers. DNS servers are connected this way hierarchically, with each server having its root servers to which it can send resolution queries. The root servers are designated in the hint zone.

Note On Red Hat you can use bindconf, the BIND Configuration Tool, to configure a DNS server for a simple local network. bindconf provides a Gnome interface for setting up the master, slave, forward, and IN-ADDR.ARPA zones you would need for a server. Be aware though that it will overwrite your **/etc/named.conf** file. bindconf can be accessed from the Gnome System menu.

DNS Servers

There are several kinds of DNS server, each designed to perform a different type of task under the Domain Name Service. The basic kind of DNS server is the *master* server. Each network must have at least one master server that is responsible for resolving names on the network. Large networks may need several DNS servers. Some of these can be *slave* servers that can be updated directly from a master server. Others may be *alternative master* servers that hosts in a network can use. Both are commonly referred to as *secondary* servers. For DNS requests a DNS server cannot resolve, the request can be forwarded to specific DNS servers outside the network, say on the Internet. DNS servers in a network can be set up to perform this task and are referred to as *forwarder* servers. To help bear the workload, local DNS servers can be set up within a network that operate as caching servers. Such a server merely collects DNS lookups from previous requests it sent to the main DNS server. Any repeated requests can then be answered by the caching server.

A server that can answer DNS queries for a given zone with authority is known as an *authoritative* server. An authoritative server holds the DNS configuration records for hosts in a zone that will associate each host's DNS name with an IP address. For example, a master server is an authoritative server. So are slave and stealth servers (see the list that follows). A caching server is not authoritative. It only holds whatever associations it picked up from other servers and cannot guarantee that the associations are valid.

- **Master server** This is the primary DNS server for a zone.

- **Slave server** A DNS server that receives zone information from the master server.
- **Forwarder server** A server that forwards unresolved DNS requests to outside DNS servers. Can be used to keep other servers on a local network hidden from the Internet.
- **Caching only server** Caches DNS information it receives from DNS servers and uses it to resolve local requests.
- **Stealth server** A DNS server for a zone not listed as a name server by the master DNS server.

Note As an alternative to making entries in the configuration files manually, you can configure DNS with Linuxconf or Webmin.

named.conf

The configuration file for the **named** daemon is **named.conf**, located in the /**etc** directory. It uses a flexible syntax similar to C programs. The format enables easy configuration of selected zones, enabling features such as access control lists and categorized logging. The **named.conf** file consists of BIND configuration commands with attached blocks within which specific options are listed. A configuration command is followed by arguments and a block that is delimited with braces. Within the block are lines of option and feature entries. Each entry is terminated with a semicolon. Comments can use the C, C++, or Shell/Perl syntax: enclosing **/* */**, preceding //, or preceding **#**. The following example shows a **zone** statement followed by the zone name and a block of options that begin with an opening brace, **{**. Each option entry ends with a semicolon. The entire block ends with a closing brace, also followed by a semicolon. The format for a **named.conf** entry is show here, along with the different kinds of comments allowed. Table 25-4 lists several commonly used options.

```
// comments
/* comments */
# comments

statements {
 options and features; //comments
};
```

The following example shows a simple caching server entry.

```
// a caching only nameserver config
//
zone "." {
       type hint;
       file "named.ca";
       };
```

Table 25-4: Zone Options

Option	Description
type	Specifies a zone type.
file	Specifies the zone file for the zone.
directory	Specifies a directory for zone files.
forwarders	Lists hosts for DNS servers where requests are to be forwarded.
masters	Lists hosts for DNS master servers for a slave server.

Table 25-4: Zone Options

Option	Description
notify	Allows master servers to notify their slave servers when the master zone data changes and updates are needed.
allow-transfer	Specifies which hosts are allowed to receive zone transfers.
allow-query	Specifies hosts that are allowed make queries.
allow-recursion	Specifies hosts that are allowed to perform recursive queries on the server.

Note The **named.conf** file is a new feature implemented with BIND version 8.x and 9.x. The older BIND 4.x versions use a file called **named.boot**. This file is no longer used by version 8.x. The syntaxes used in these configuration files differ radically. If you upgrade to 8.x, you can use the **named-bootconf.pl** Perl script provided with the BIND software to convert your **named.boot** file to a **named.conf** file.

The **zone** statement is used to specify the domains the name server will service. You enter the keyword **zone,** followed by the name of the domain placed within double quotes. Do not place a period at the end of the domain name. In the following example, a period is within the domain name, but not at the end, **"mytrek.com";** this differs from the zone file, which requires a period at the end of a complete domain name.

After the zone name, you can specify the class **in**, which stands for Internet. You can also leave it out, in which case **in** is assumed (there are only a few other esoteric classes that are rarely used). Within the zone block, you can place several options (see Table 25-4). Two essential options are **type** and **file**. The **type** option is used to specify the zone's type. The **file** option is used to specify the name of the zone file to be used for this zone. You can choose from several types of zones: master, slave, stub, forward, and hint. *Master* specifies that the zone holds master information and is authorized to act on it. A master server was called a primary server in the older 4.x BIND configuration. *Slave* indicates that the zone needs to update its data periodically from a specified master name server. You use this entry if your name server is operating as a secondary server for another primary (master) DNS server. A *stub zone* only copies other name server entries, instead of the entire zone. A *forward zone* directs all queries to name servers specified in a **forwarders** statement. A *hint zone* specifies the set of root name servers used by all Internet DNS servers. You can also specify several options that can override any global options set with the **options** statement. Table 25-5 lists the BIND zone types. The following example shows a simple **zone** statement for the **mytrek.com** domain. Its class is Internet, "in," and its type is master. The name of its zone file is usually the same as the zone name, in this case, **"mytrek.com"**.

```
zone "mytrek.com" in {
       type master;
       file "mytrek.com";
       };
```

Table 25-5: DNS BIND Zone Types

Type	Description
master	Primary DNS zone
slave	Slave DNS server. Controlled by a master DNS server
hint	Set of root DNS Internet servers

Table 25-5: DNS BIND Zone Types

Type	Description
forward	Forwards any queries in it to other servers
stub	Like a slave zone, but only holds names of DNS servers

Other statements, such as **acl**, **server**, **options**, and **logging**, enable you to configure different features for your name server (see Table 25-6). The **server** statement defines the characteristics to be associated with a remote name server, such as the transfer method and key ID for transaction security. The **control** statement defines special control channels. The **key** statement defines a key ID to be used in a **server** statement that associates an authentication method with a particular name server (see DNSSEC). The **logging** statement is used to configure logging options for the name server, such as the maximum size of the log file and a severity level for messages. Table 25-6 lists the BIND statements.

Table 25-6: BIND Configuration Statements

Statement	Description
/* *comment* */	BIND comment in C syntax.
// *comment*	BIND comment in C++ syntax.
# *comment*	BIND comment in Unix shell and Perl syntax.
acl	Defines a named IP address matching list.
include	Includes a file, interpreting it as part of the **named.conf** file.
key	Specifies key information for use in authentication and authorization.
logging	Specifies what the server logs and where the log messages are sent.
>Option	Global server configuration options and defaults for other statements.
controls	Declares control channels to be used by the ndc utility.
server	Sets certain configuration options for the specified server basis.
sortlists	Gives preference to specified networks based on a queries source.
trusted-keys	Defines DNSSEC keys preconfigured into the server and implicitly trusted.
zone	Defines a zone.
view	Defines a view.

The **options** statement defines global options and can be used only once in the configuration file. An extensive number of options cover such components as forwarding, name checking, directory path names, access control, and zone transfers, among others (see Table 25-7). A complete listing can be found in the BIND documentation. A critically important option found in most configuration files is the **directory** option, which holds the location of the name server's zone and cache files on your system. The following example is taken from the Red Hat **/etc/named.conf** file. This example specifies the zone files are located in the **/var/named** directory. In this directory, you can find your zone files, including those used for your local system.

```
options {
        directory "/var/named";
        forwarders { 192.168.1.34;
                    192.168.1.47;
                  };
      };
```

Table 25-7: options Options

Option	**Description**
sortlist	Gives preference to specified networks based on a queries source.
directory	Specifies a directory for zone files.
forwarders	Lists hosts for DNS servers where requests are to be forwarded.
allow-transfer	Specifies which hosts are allowed to receive zone transfers.
allow-query	Specifies hosts that are allowed make queries.
allow-recursion	Specifies hosts that are allowed to perform recursive queries on the server.
notify	Allows master servers to notify their slave servers when the master zone data changes and updates are needed.
blackhole	Option to eliminate denial response by **allow-query**.

Another commonly used global option is the **forwarders** option. With the **forwarders** option, you can list several DNS servers to which queries can be forwarded if they cannot be resolved by the local DNS server. This is helpful for local networks that may need to use a DNS server connected to the Internet. The **forwarders** option can also be placed in forward zone entries.

With the **notify** option turned on, the master zone DNS servers send messages to any slave DNS servers whenever their configuration has changed. The slave servers can then perform zone transfers in which they download the changed configuration files. Slave servers always use the DNS configuration files copied from their master DNS servers. **notify** takes one argument, **yes** or **no**, where **yes** is the default. With the **no** argument, you can have the master server not send out any messages to the slave servers, in effect preventing any zone transfers.

The **sortlists** statement lets you specify preferences to be used when a query returns multiple responses. For example, you could give preference to your localhost network or to a private local network such a 192.168.1.0.

The following example is a simple **named.conf** file based on the example provided in the BIND documentation. This example shows samples of several of the configuration statements. The file begins with comments using C++ syntax, //. The **options** statement has a directory entry that sets the directory for the zone and cache files to **/var/named**. Here, you find your zone files, such as **named.local** and reverse mapping files, along with the cache file, **named.ca**. The first **zone** statement (**.**) defines a hint zone specifying the root name servers. The cache file listing these servers is **named.ca**. The second **zone** statement defines a zone for the **mytrek.com** domain. Its type is master, and its zone file is named **"mytrek.com"**. The next zone is used for reverse IP mapping of the previous zone. Its name is made up of a reverse listing of the **mytrek.com** domain's IP address with the term **IN-ADDR.ARPA** appended. The domain address for **mytrek.com** is 192.168.1, so the reverse is 1.168.192. The **IN-ADDR.ARPA** domain is a special domain that supports gateway location and Internet address to host mapping. The last **zone** statement defines a reverse mapping zone for the loopback interface, the method used by the system to address itself and enable communication between local users on the system. The zone file used for this local zone is **named.local**.

named.conf

```
//
// A simple BIND 9 configuration
//

logging {
        category cname { null; };
        };

options {
        directory "/var/named";
        };

zone "." {
         type hint;
         file "named.ca";
         };

zone "mytrek.com" {
               type master;
               file "mytrek.com";
               };
zone "1.168.192.IN-ADDR.ARPA" {
                 type master;
                 file "192.168.1";
                 };

zone "0.0.127.IN-ADDR.ARPA" {
                 type master;
                 file "named.local";
                 };
```

When BIND is initially installed, it creates a default configuration for what is known as a caching only server. A *caching only server* copies queries made by users and saves them in a cache, for use later if the queries are repeated. This can save DNS lookup response times. The cache is held in memory and only lasts as long as **named** runs. The following example is the **named.conf** file initially installed for a caching only server. Only the local and cache zones are defined.

named.conf (caching only server)

```
// generated by named-bootconf.pl

options {
        directory "/var/named";
        };

//
// a caching only nameserver config
//
zone "." {
```

```
          type hint;
         file "named.ca";
         };

zone "0.0.127.IN-ADDR.ARPA" {
                 type master;
                 file "named.local";
                 };
```

Resource Records

Your name server holds domain name information about the hosts on your network in resource records placed in zone and reverse mapping files. Resource records are used to associate IP addresses with fully qualified domain names. You need a record for every computer in the zone that the name server services. A record takes up one line, though you can use parentheses to use several lines for a record, as is usually the case with SOA records. A resource record uses the Standard Resource Record Format as shown here:

```
name [<ttl>] [<class>] <type> <rdata> [<comment>]
```

Here, *name* is the name for this record. It can be a domain name for a fully qualified domain name. If you only specify the hostname, the default domain is appended. If no name entry exists, the last specific name is used. If the **@** symbol is used, the name server's domain name is used. *ttl* (time to live) is an optional entry that specifies how long the record is to be cached, and *class* is the class of the record. The class used in most resource record entries is IN, for Internet. By default, it is the same as that specified for the domain in the **named.conf** file. *type* is the type of the record. *rdata* is the resource record data. The following is an example of a resource record entry. The name is **rabbit.mytrek.com**, the class is Internet (IN), the type is a host address record (A), and the data is the IP address 192.168.1.2.

```
rabbit.mytrek.com. IN A 192.168.1.2
```

Table 25-8: Domain Name Service Resource Record Types

Type	**Description**
A	Host address, maps host name to IP address
A6	An IPv6 Host address
NS	Authoritative name server for this zone
CNAME	Canonical name, used to define an alias for a hostname
SOA	Start of Authority, starts DNS entries in zone file, specifies name server for domain, and other features like server contact and serial number
WKS	Well-known service description
PTR	Pointer record, for performing reverse domain name lookups, maps IP address to hostname
RP	Text string that contains contact information about a host
HINFO	Host information
MINFO	Mailbox or mail list information
MX	Mail exchanger, informs remote site of your zone's mail server
TXT	Text strings, usually information about a host

Table 25-8: Domain Name Service Resource Record Types

Type	**Description**
KEY	Domain private key
SIG	Resource record signature
NXT	Next resource record

Different types of resource records exist for different kinds of hosts and name server operations (see Table 25-8 for a listing of resource record types). A, NS, MX, PTR, and CNAME are the types commonly used. *A* is used for host address records that match domain names with IP addresses. NS is used to reference a name server. MX specifies the host address of the mail server that services this zone. The name server has mail messages sent to that host. The PTR type is used for records that point to other resource records and is used for reverse mapping. CNAME is used to identify an alias for a host on your system.

Start of Authority: SOA

A zone and reverse mapping files always begin with a special resource record called the Start of Authority (SOA) record. This record specifies that all the following records are authoritative for this domain. It also holds information about the name server's domain, which is to be given to other name servers. An SOA record has the same format as other resource records, though its data segment is arranged differently. The format for an SOA record follows:

```
name {ttl} class SOA Origin Person-in-charge (
                                Serial number
                                Refresh
                                Retry
                                Expire
                                Minimum )
```

Each zone has its own SOA record. The SOA begins with the zone name specified in the **named.conf** zone entry. This is usually a domain name. An @ symbol is usually used for the name and acts like a macro expanding to the domain name. The *class* is usually the Internet class, IN. *SOA* is the type. *Origin* is the machine that is the origin of the records, usually the machine running your name server daemon. The *person-in-charge* is the e-mail address for the person managing the name server (use dots, not @, for the e-mail address, as this symbol is used for the domain name). Several configuration entries are placed in a block delimited with braces. The first is the *serial number.* You change the serial number when you add or change records, so that it is updated by other servers. The serial number can be any number, as long as it is incremented each time a change is made to any record in the zone. A common practice is to use the year-month-day number for the serial number, where number is the number of changes in that day. For example, 1999120403 would be the year 1999, December 4, for the third change. Be sure to update it when making changes.

Refresh specifies the time interval for refreshing SOA information. *Retry* is the frequency for trying to contact an authoritative server. *Expire* is the length of time a secondary name server keeps information about a zone without updating it. *Minimum* is the length of time records in a zone live. The times are specified in the number of seconds.

The following example shows an SOA record. The machine running the name server is **turtle.mytrek.com,** and the e-mail address of the person responsible for the server is **hostmaster.turtle.mytrek.com**. Notice the periods at the end of these names. For names with no periods, the domain name is appended. **turtle** would be the same as **turtle.mytrek.com**. When entering full hostnames, be sure to add the period so that the domain is not appended.

```
@ IN SOA turtle.mytrek.com. hostmaster.turtle.mytrek.com. (
                                1997022700 ; Serial
                                28800 ; Refresh
                                14400 ; Retry
                                3600000 ; Expire
                                86400 ) ; Minimum
```

Name Server: NS

The name server record specifies the name of the name server for this zone. It has a resource record type of NS. If you have more than one name server, list them in NS records. These records usually follow the SOA record. As they usually apply to the same domain as the SOA record, their name field is often left blank to inherit the server's domain name specified by the **@** symbol in the previous SOA record.

```
              IN   NS      turtle.mytrek.com.
```

You can, if you wish, enter the domain name explicitly as shown here:

```
mytrek.com.   IN   NS      turtle.mytrek.com.
```

Address Record: A and A6

Resource records of type A are address records that associate a fully qualified domain name with an IP address. Often, only their hostname is specified. Any domain names without a terminating period automatically have the domain appended to it. Given the domain **mytrek.com**, the turtle name in the following example is expanded to **turtle.mytrek.com**:

```
rabbit.mytrek.com. IN     A      192.168.1.2
turtle             IN     A      192.168.1.1
```

BIND versions 8.2.2 and 9.1 support IPv6 addresses. IPv6 IP addresses have a very different format from that of the IPv4 addresses commonly used (see Chapter 38). Instead of the numerals arranged in four segments, IPv6 uses hexadecimal numbers arranged in seven segments. Though BIND checks for both IPv4 and IPv6 addresses, currently you should always use a system's IPv4 address if it has one. In the following example, **divit.mygolf.com** is associated with its IPv6 address.

```
divit.mygolf.com    IN    A6    3ffe:8050:201:1860:1::3
```

BIND also supports IPv6 resolution features such as A6 chains, which allows you to specify part of the address as a network domain name. This would be the name of a network through which the host connects to the Internet. The network domain name is then used to complete the address.

```
divit.mygolf.com    IN    A6    0:0:0:0:1::3mytrek.com.
```

Mail Exchanger: MX

The Mail Exchanger record, MX, specifies the mail server that is used for this zone or for a particular host. The mail exchanger is the server to which mail for the host is sent. In the following example, the mail server is specified as **turtle.mytrek.com**. Any mail sent to the address for any machines in that zone will be sent to the mail server, which in turn will send it to the specific machines. For example, mail sent to a user on **rabbit.mytrek.com** will first be sent to **turtle.mytrek.com**, which will then send it on to **rabbit.mytrek.com**. In the following example, the host 192.168.1.1 (**turtle.mytrek.com**) is defined as the mail server for the **mytrek.com** domain:

```
mytrek.com.  IN     MX   10   192.168.1.1
```

You could also inherit the domain name from the SOA record, leaving the domain name entry blank.

```
             IN     MX   10   192.168.1.1
```

An MX record recognizes an additional field that specifies the ranking for a mail exchanger. If your zone has several mail servers, you can assign them different rankings in their MX records. The smaller number has a higher ranking. This way, if mail cannot reach the first mail server, it can be routed to an alternate server to reach the host. In the following example, mail for hosts on the **mytrek.com** domain is first routed to the mail server at 192.168.1.1 (**turtle.mytrek.com**), and if that fails, it is routed to the mail server at 192.168.1.1 (**rabbit.mytrek.com**).

```
mytrek.com. IN MX 10 turtle.mytrek.com
            IN MX 20 rabbit.mytrek.com
```

You can also specify a mail server for a particular host. In the following example, the mail server for **lizard.mytrek.com** is specified as **rabbit.mytrek.com**:

```
lizard.mytrek.com. IN      A        192.168.1.3
                   IN      MX  10   rabbit.mytrek.com.
```

Aliases: CNAME

Resource records of type CNAME are used to specify alias names for a host in the zone. Aliases are often used for machines running several different types of servers, such as both Web and FTP servers. They are also used to locate a host when it changes its name. The old name becomes an alias for the new name. In the following example, **ftp.mytrek.com** is an alias for a machine actually called **turtle.mytrek.com**:

```
ftp.mytrek.com. IN CNAME turtle.mytrek.com.
```

The term CNAME stands for canonical name. The canonical name is the actual name of the host. In the example above the canonical name is **turtle.mytrek.com**. The alias, also know as the CNAME, is **ftp.mytrek.com**. In a CNAME entry, the alias points to the canonical name. Aliases cannot be used for NS (name server) or MX (mail server) entries.

A more stable way to implement aliases is simply to create another address record for it. You can have as many hostnames for the same IP address as you want, provided they are certified. For example, to make **www.mytrek.com** an alias for **turtle.mytrek.com**, you only have to add another address record for it, giving it the same IP address as **turtle.mytrek.com**.

```
turtle.mytrek.com. IN A 192.168.1.1
www.mytrek.com. IN A 192.168.1.1
```

Pointer Record: PTR

A PTR record is used to perform reverse mapping from an IP address to a host. PTR records are used in the reverse mapping files. The name entry holds a reversed IP address and the data entry holds the name of the host. The following example maps the IP address 192.168.1.1 to **turtle.mytrek.com**:

```
1.1.168.192 IN PTR turtle.mytrek.com.
```

In a PTR record you can specify just that last number segment of the address (the host address), and let DNS fill in the domain part of the address. In the next example, 1 has the domain address, 1.168.192, automatically added to give 1.1.168.192:

```
1 IN PTR turtle.mytrek.com.
```

Host Information: HINFO, RP, MINFO, and TXT

The HINFO, RP, MINFO, and TXT records are used to provide information about the host. The RP record enables you to specify the person responsible for a certain host. The HINFO record provides basic hardware and operating system identification. The TXT record is used to enter any text you want. MINFO provides a host's mail and mailbox information. These are used sparingly as they may give too much information out about the server.

Zone Files

A DNS server uses several zone files covering different components of the DNS. Each zone uses two zone files: the principal zone file and a reverse mapping zone file. The *zone file* contains the resource records for hosts in the zone. A *reverse mapping file* contains records that provide reverse mapping of your domain name entries, enabling you to map from IP addresses to domain names. The name of the file used for the zone file can be any name. The name of the file is specified in the **zone** statement's file entry in the **named.conf** file. If your server supports several zones, you may want to use a name that denotes the specific zone. Most systems use the domain name as the name of the zone file. For example, the zone **mytrek.com** would have a zone file also called **mytrek.com**. These could be placed in a subdirectory called **zones** or **master**. The zone file used in the following example is called **mytrek.com**. The reverse mapping file can also be any name, though it is usually the reverse IP address domain specified in its corresponding zone file. For example, in the case of **mytrek.com** zone file, the reverse mapping file might be called **192.168.1**, the IP address of the **mytrek.com** domain defined in the **mytrek.com** zone file. This file would contain reverse mapping of all the host addresses in the domain, allowing their hostname addresses to be mapped to their corresponding IP addresses. In addition, BIND sets up a cache file and a reverse mapping file for the localhost. The cache file holds the resource records for the root name servers to which your name server connects. The cache file can be any name, although it is usually called **named.ca**. The localhost reverse mapping file holds reverse IP resource records for the local loopback interface, localhost. Although localhost can be any name, it usually has the name **named.local**.

Zone Files for Internet Zones

A zone file holds resource records that follow a certain format. The file begins with general directives to define default domains or to include other resource record files. These are followed by a single SOA, name server, and domain resource records, and then resource records for the different hosts. Comments begin with a semicolon and can be placed throughout the file. The @ symbol operates like a special macro, representing the domain name of the zone to which the records apply. The @ symbol is used in the first field of a resource or SOA record as the zone's domain name. Multiple names can be specified using the * matching character. The first field in a resource record is the name of the domain to which it applies. If the name is left blank, the next previous explicit name entry in another resource record is automatically used. This way, you can list several entries that apply to the same host without having to repeat the host name. Any host or domain name used throughout this file that is not terminated with a period has the zone's domain appended to it. For example, if the zone's domain is **mytrek.com** and a resource record has only the name **rabbit** with no trailing period, the zone's domain is automatically appended to it, giving you **rabbit.mytrek.com.** Be sure to include the trailing period whenever you enter the complete fully qualified domain name as in **turtle.mytrek.com.**. You can also use several directives to set global attributes. $ORIGIN sets a default domain name to append to address names that do not end in a period. $INCLUDE includes a file. $GENERATE can generate records whose domain or IP addresses differ only by an iterated number.

A zone file begins with an SOA record specifying the machine the name server is running on, among other specifications. The @ symbol is used for the name of the SOA record, denoting the zone's domain name. After the SOA, the name server resource records (NS) are listed. Just below the name server records are resource records for the domain itself. Resource records for host addresses (A), aliases (CNAME), and mail exchangers (MX) follow. The following example shows a sample zone file, which begins with an SOA record and is followed by an NS record, resource records for the domain, and then resource records for individual hosts:

```
; Authoritative data for turle.mytrek.com
;
@ IN SOA turtle.mytrek.com. hostmaster.turtle.mytrek.com.(
                             93071200 ; Serial number
                                10800 ; Refresh 3 hours
                                 3600 ; Retry 1 hour
                              3600000 ; Expire 1000 hours
                              86400 ) ; Minimum 24 hours

             IN      NS         turtle.mytrek.com.
             IN      A          192.168.1.1
             IN      MX   10    turtle.mytrek.com.
             IN      MX   15    rabbit.mytrek.com.

turtle       IN      A          192.168.1.1
             IN      HINFO      PC-686 LINUX
gopher       IN      CNAME      turtle.mytrek.com.
ftp          IN      CNAME      turtle.mytrek.com.
www          IN      A          192.168.1.1
```

```
rabbit        IN      A         192.168.1.2

lizard        IN      A         192.168.1.3
              IN      HINFO     MAC MACOS
localhost     IN      A         127.0.0.1
```

The first two lines are comments about the server for which this zone file is used. Notice that the first two lines begin with a semicolon. The class for each of the resource records in this file is IN, indicating these are Internet records. The SOA record begins with an @ symbol that stands for the zone's domain. In this example, it is **mytrek.com**. Any host or domain name used throughout this file that is not terminated with a period has this domain appended to it. For example, in the following resource record, **turtle** has no period, so it automatically expands to **turtle.mytrek.com**. The same happens for **rabbit** and **lizard**. These are read as **rabbit.mytrek.com** and **lizard.mytrek.com**. Also, in the SOA, notice that the e-mail address for host master uses a period instead of an @ symbol; @ is a special symbol in zone files and cannot be used for any other purpose.

The next resource record specifies the name server for this zone. Here, it is **mytrek.com.** Notice the name for this resource record is blank. If the name is blank, a resource record inherits the name from the previous record. In this case, the NS record inherits the value of @ in the SOA record, its previous record. This is the zone's domain and the NS record specifies **turtle.mytrek.com** is the name server for this zone.

```
              IN   NS    turtle.mytrek.com.
```

Here, the domain name is inherited. The entry can be read as the following. Notice the trailing period at the end of the domain name.

```
 mytrek.com.  IN   NS    turtle.mytrek.com.
```

The following address records set up an address for the domain itself. This is often the same as the name server, in this case 192.168.1.1 (the IP address of **turtle.mytrek.com**). This enables users to reference the domain itself, rather than a particular host in it. A mail exchanger record follows that routes mail for the domain to the name server. Users can send mail to the **mytrek.com** domain, and it will be routed to **turtle.mytrek.com.**

```
                IN    A     192.168.1.1
```

Here the domain name is inherited. The entry can be read as the following.

```
 mytrek.com.    IN    A     192.168.1.1
```

The next records are mail exchanger records (MX) listing **turtle.mytrek.com** and **fast.mytrek.com** as holding the mail servers for this zone. You can have more than one mail exchanger record for host. More than one host may exist through which mail can be routed. These can be listed in mail exchanger records for which you can set priority rankings (smaller number ranks higher). In this example, if **turtle.mytrek.com** cannot be reached, its mail is routed through **rabbit.mytrek.com**, which has been set up also to handle mail for the **mytrek.com** domain.

```
          IN     MX    100    turtle.mytrek.com.
          IN     MX    150    rabbit.mytrek.com.
```

Again the domain name is inherited. The entries can be read as the following.

```
mytrek.com.       IN      MX  100   turtle.mytrek.com.
mytrek.com.       IN      MX  150   rabbit.mytrek.com.
```

The following resource record is an address record (A) that associates an IP address with the fully qualified domain name **turtle.mytrek.com**. The resource record name only holds **turtle** with no trailing period, so it is automatically expanded to **turtle.mytrek.com.** This record provides the IP address to which **turtle.mytrek.com** can be mapped.

```
turtle   IN    A     192.168.1.1
```

Several resource records immediately follow that have blank names. These inherit their names from the preceding full record-in this case, **turtle.mytrek.com.** In effect, these records also apply to that host. Using blank names is an easy way to list additional resource records for the same host (notice that an apparent indent occurs). The first record is an information record, providing the hardware and operating system for the machine.

```
         IN    HINFO    PC-686 LINUX
```

If you are using the same machine to run several different servers, such as Web, FTP, and Gopher servers, you may want to assign aliases to these servers to make accessing them easier for users. Instead of using the actual domain name, such as **turtle.mytrek.com,** to access the Web server running on it, users may find using the following is easier: for the Web server, **www.mytrek.com;** for the Gopher server, **gopher.mytrek.com;** and for the FTP server, **ftp.mytrek.com**. In the DNS, you can implement such a feature using alias records. In the example zone file, two CNAME alias records exist for the **turtle.mytrek.com** machine: FTP and Gopher. The next record implements an alias for **www** using another address record for the same machine. None of the name entries end in a period, so they are appended automatically with the domain name **mytrek.com**. **www.mytrek.com**, **ftp.mytrek.com**, and **gopher.mytrek.com** are all aliases for **turtle.mytrek.com**. Users entering those URLs automatically access the respective servers on the **turtle.mytrek.com** machine.

Address and main exchanger records are then listed for the two other machines in this zone: **rabbit.mytrek.com** and **lizard.mytrek.com**. You could add HINFO, TXT, MINFO, or alias records for these entries. The file ends with an entry for localhost, the special loopback interface that allows your system to address itself.

Reverse Mapping File

Reverse name lookups are enabled using a reverse mapping file. *Reverse mapping files* map fully qualified domain names to IP addresses. This reverse lookup capability is unnecessary, but it is convenient to have. With reverse mapping, when users access remote hosts, their domain name address can be used to identify their own host, instead of only the IP address. The name of the file can be anything you want. On most current distributions, it is the zone's domain address (the network part of a zone's IP address). For example, the reverse mapping file for a zone with the IP address of 192.168.1.1 is 192.168.1. Its full pathname would be something like **/var/named/192.168.1**. On some systems using older implementations of BIND, the reverse mapping filename may consist of the root name of the zone file with the extension **.rev**. For example, if the zone file is called **mytrek.com,** the reverse mapping file would be called something like **mytrek.rev**. The zone entry for a reverse mapping in the

named.conf file uses a special domain name consisting of the IP address in reverse, with an **IN-ADDR.ARPA** extension. This reverse IP address becomes the zone domain referenced by the @ symbol in the reverse mapping file. For example, the reverse mapping zone name for a domain with the IP address of **192.168.43** would be **43.168.192.IN-ADDR.ARPA**. In the following example, the reverse domain name for the domain address **192.168.1** is **1.168.192.IN-ADDR.ARPA**:

```
zone "1.168.192.IN-ADDR.ARPA" in {
          type master;
          file "192.168.1";
          };
```

A reverse mapping file begins with an SOA record, which is the same as that used in a forward mapping file. Resource records for each machine defined in the forward mapping file then follow. These resource records are PTR records that point to hosts in the zone. These must be actual hosts, not aliases defined with CNAME records. Records for reverse mapping begin with a reversed IP address. Each segment in the IP address is sequentially reversed. Each segment begins with the host ID, followed by reversed network numbers. If you list only the host ID with no trailing period, the zone domain is automatically attached. In the case of a reverse mapping file, the zone domain as specified in the **zone** statement is the domain IP address backward. The 1 expands to 1.1.168.192. In the following example, **turtle** and **lizard** inherit the domain IP address, whereas **rabbit** has its explicitly entered:

```
; reverse mapping of domain names 1.168.192.IN-ADDR.ARPA
;
@ IN SOA turtle.mytrek.com. hostmaster.turtle.mytrek.com.(
                        92050300 ; Serial (yymmddxx format)
                           10800 ; Refresh 3hHours
                            3600 ; Retry 1 hour
                         3600000 ; Expire 1000 hours
                         86400 ) ; Minimum 24 hours

@            IN    NS       turtle.mytrek.com.
1            IN    PTR      turtle.mytrek.com.
2.1.168.192  IN    PTR      rabbit.mytrek.com.
3            IN    PTR      lizard.mytrek.com.
```

Localhost Reverse Mapping

A localhost reverse mapping file implements reverse mapping for the local loopback interface known as *localhost,* whose network address is 127.0.0.1. This file can be any name. On most systems, localhost is given the name **named.local**. On other systems, localhost may use the network part of the IP address, 127.0.0. This file allows mapping the domain name localhost to the localhost IP address, which is always 127.0.0.1 on every machine. The address 127.0.0.1 is a special address that functions as the local address for your machine. It allows a machine to address itself. In the **zone** statement for this file, the name of the zone is **0.0.127.IN-ADDR.ARPA**. The domain part of the IP address is entered in reverse order, with **IN-ADDR.ARPA** appended to it, **0.0.127.IN- ADDR.ARPA**. The **named.conf** entry is shown here:

```
zone "0.0.127.IN-ADDR.ARPA" {
           type master;
           file "named.local";
           };
```

The name of the file used for the localhost reverse mapping file is usually **named.local**, though it can be any name. The NS record specifies the name server localhost should use. This file has a PTR record that maps the IP address to the localhost. The 1 used as the name expands to append the zone domain-in this case, giving you 1.0.0.127, a reverse IP address. The contents of the **named.local** file are shown here. Notice the trailing periods for localhost.

```
@ IN SOA localhost. root.localhost. (
                        1997022700 ; Serial
                             28800 ; Refresh
                             14400 ; Retry
                           3600000 ; Expire
                           86400 ) ; Minimum

         IN      NS    turtle.mytrek.com.
1        IN      PTR   localhost.
```

Subdomains and Slaves

Adding a subdomain to a DNS server is a simple matter of creating an additional master entry in the **named.conf** file, and then placing name server and authority entries for that subdomain in your primary DNS server's zone file. The subdomain, in turn, has its own zone file with its SOA record and entries listing hosts, which are part of its subdomain, including any of its own mail and news servers.

The name for the subdomain could be a different name altogether or a name with the same suffix as the primary domain. In the following example, the subdomain is called **beach.mytrek.com**. It could just as easily be called **mybeach.com**. The name server to that domain is on the host **crab.beach.mytrek.com**, in this example. Its IP address is 192.168.1.33 and its zone file is **beach.mytrek.com**. The **beach.mytrek.com** zone file holds DNS entries for all the hosts being serviced by this name server. The following example shows zone entries for its **named.conf**:

```
zone "beach.mytrek.com" {
          type master;
          file "beach.mytrek.com";
          };

zone "1.168.192.IN-ADDR.ARPA" {
          type master;
          file "192.168.1";
          };
```

On the primary DNS server, in the example **turtle.mytrek.com**, you would place entries in the master zone file to identify the subdomain server's host and designate it as a name server. Such entries are also known as glue records. In this example, you would place the following entries in the **mytrek.com** zone file on **turtle.mytrek.com**:

```
beach.mytrek.com.      IN    NS    beach.mytrek.com.
beach.mytrek.com.      IN    A     192.168.1.33.
```

URL references to hosts serviced by **beach.mytrek.com** can now be reached from any host serviced by **mytrek.com**, which does not need to maintain any information about the **beach.mytrek.com** hosts. It simply refers such URL references to the **beach.mytrek.com** name server.

A slave DNS server is tied directly to a master DNS server and periodically receives DNS information from it. You use a master DNS server to configure its slave DNS servers automatically. Any changes you make to the master server are automatically transferred to its slave servers. This transfer of information is called a *zone transfer.* Zone transfers are automatically initiated whenever the slave zone's refresh time is reached or the slave server receives a notify message from the master. The *refresh time* is the second argument in the zone's SOA entry. A notify message is automatically sent by the master whenever changes are made to the master zone's configuration files and the **named** daemon is restarted. In effect, slave zones are automatically configured by the master zone, receiving the master zone's zone files and making them their own.

Using the previous examples, suppose you want to set up a slave server on **rabbit.mytrek.com**. Zone entries, as shown in the following example, are set up in the **named.conf** configuration file for the slave DNS server on **rabbit.mytrek.com**. The slave server is operating in the same domain as the master, and so it has the same zone name, **mytrek.com**. Its SOA file is named **slave.mytrek.com**. The term "slave" in the filename is merely a convention that helps identify it as a slave server configuration file. The **masters** statement lists its master DNS server-in this case, 192.168.1.1. Whenever the slave needs to make a zone transfer, it transfers data from that master DNS server. The entry for the reverse mapping file for this slave server lists its reverse mapping file as **slave.192.168.1**.

```
zone "mytrek.com" {
         type slave;
         file "slave.mytrek.com";
         masters { 192.168.1.1;
         };

zone "1.168.192.IN-ADDR.ARPA" {
          type slave;
          file "slave.192.168.1";
          masters { 192.168.1.1;
          };
```

On the master DNS server, the master SOA zone file has entries in it to identify the host that holds the slave DNS server and to designate it as a DNS server. In this example, you would place the following in the **mytrek.com** zone file:

```
         IN       NS       192.168.1.2
```

You would also place an entry for this name server in the **mytrek.com** reverse mapping file:

```
         IN       NS       192.168.1.2
```

The master DNS server can control which slave servers can transfer zone information from it using the **allow-transfer** statement. Place the statement with the list of IP addresses for the slave servers for which you want to allow access. Also, the master DNS server should be sure the **notify** option is not disabled. The **notify** option is disabled by a "notify no" statement in the options or zone **named.conf** entries. Simply erase the "no" argument to enable notify.

With BIND versions 8.2.2 and 9.0, BIND now supports Incremental Zone Transfers (IXFR). Previously the all zone data would be replaced in an update, rather than simply editing in changes such as the addition of a few resource records. With Incremental Zone Transfers, a database of changes is maintained by the master zone. Then only the changes are transferred to the slave zone, which uses this information to update its own zone files. To implement Incremental Zone Transfers, you have to turn on the **maintain- ixfr-base** option in the options section.

```
maintain-ixfr-base yes;
```

You can then use the **ixfr-base** option in a zone section to specify a particular database file to hold changes.

```
ixfr-base "db.mytrek.com.ixfr";
```

IP Virtual Domains

IP-based virtual hosting allows more than one IP address to be used for a single machine. If a machine has two registered IP addresses, either one can be used to address the machine. If you want to treat the extra IP address as another host in your domain, you need only create an address record for it in your domain's zone file. The domain name for the host would be the same as your domain name. If you want to use a different domain name for the extra IP, however, you have to set up a virtual domain for it. This entails creating a new **zone** statement for it with its own zone file. For example, if the extra IP address is 192.168.1.42 and you want to give it the domain name **sail.com**, you must create a new **zone** statement for it in your **named.conf** file with a new zone file. The **zone** statement would look something like this. The zone file is called **sail.com**.

```
zone "sail.com" in {
           type master;
           file "sail.com";
           };
```

In the "**sail.com**" file, the name server name is **turtle.mytrek.com** and the e-mail address is **hostmaster@turtle.mytrek.com**. In the name server (NS) record, the name server is **turtle.mytrek.com.** This is the same machine using the original address that the name server is running as. **turtle.mytrek.com** is also the host that handles mail addressed to **sail.com** (MX). An address record then associates the extra IP address 192.168.1.42 with the **sail.com** domain name. A virtual host on this domain is then defined as **jib.sail.com**. Also, **www** and **ftp** aliases are created for that host, creating **www.sail.com** and **ftp.sail.com** virtual hosts.

```
; Authoritative data for sail.com
;
@ IN SOA turtle.mytrek.com. hostmaster.turtle.mytrek.com. (
                                93071200 ; Serial (yymmddxx)
                                   10800 ; Refresh 3 hours
                                    3600 ; Retry 1 hour
                                 3600000 ; Expire 1000 hours
                                 86400 ) ; Minimum 24 hours

        IN      NS         turtle.mytrek.com.
        IN      MX   10    turtle.mytrek.com.
        IN      A          192.168.1.42 ;address of the sail.com domain

jib     IN      A         192.168.1.42
www     IN      A         jib.sail.com.
ftp     IN      CNAME     jib.sail.com.
```

In your reverse mapping file (**/var/named/1.168.192**), add PTR records for any virtual domains.

```
42.1.168.192       IN      PTR     sail.com.
42.1.168.192       IN      PTR     jib.sail.com.
```

You also have to configure your network connection to listen for both IP addresses on your machine (see Chapter 28).

Cache File

The *cache file* is used to connect the DNS server to root servers on the Internet. The file can be any name. On many systems, the cache file is called **named.ca**. Other systems may call the cache file **named.cache** or **roots.hints**. The cache file is usually a standard file installed by your BIND software, which lists resource records for designated root servers for the Internet. You can obtain a current version of the **named.ca** file from the **rs.internic.net** FTP site. The following example shows sample entries taken from the **named.ca** file:

```
; formerly NS.INTERNIC.NET
;
. 3600000 IN NS A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET. 3600000 A 198.41.0.4
;
; formerly NS1.ISI.EDU
;
. 3600000 NS B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET. 3600000 A 128.9.0.107
```

If you are creating an isolated intranet, you need to create your own root DNS server until you connect to the Internet. In effect, you are creating a fake root server. This can be another server on your system pretending to be the root or the same name server.

DNS Security: Access Control Lists and DNSSEC

DNS security currently allows you to control specific access by hosts to the DNS server, as well as providing encrypted communications between servers. With access control lists, you can determine who will have access to your DNS server. The DNS Security Extensions (DNSSEC), included with BIND 9.x, provide private/public key encrypted authentication and transmissions.

Access Control Lists

To control access by other hosts, you use access control lists, implemented with the **acl** statement. **allow** and **deny** options with access control host lists enable you to deny or allow access by specified hosts to the name server. With **allow-query** you can restrict queries to specified hosts or networks. Normally this will result in a response saying that access is denied. You can further eliminate this response by using the **blackhole** option in the **options** statement.

You define an acl list with the **acl** statement followed by the label you want to give the list and then the list of addresses. Addresses can be IP addresses, network addresses, or a range of addresses based on CNDR notation. You can also use an acl list defined earlier. The following example defines an acl list called **mynet**:

```
acl mynet { 192.168.1.1; 192.168.1.2; };
```

If you are specifying a range, like a network, you also add exceptions to the list by preceding such addresses with an **!**. In the following example, the mynetx acl lists all those in the 192.168.1.0 network, except for 192.168.1.3:

```
acl myexceptions {192.168.1.0; !192.168.1.3; };
```

Four default acl lists are already defined for you. You can use them wherever an option uses a list of addresses as an argument. These are **any** for all hosts, **none** for no hosts, **localhost** for all local IP addresses, and **localnet** for all hosts on local networks served by the DNS server.

Once a list is defined, you can then use it with the **allow-query**, **allow-transfer**, **allow-recursion**, and **blackhole** options in a **zone** statement to control access to a zone. **allow-query** specifies hosts that can query the DNS server. **allow-transfer** is used for master/slave zones, designating whether update transfers are allowed. **allow-recursion** specifies those hosts that can perform recursive queries on the server. The **blackhole** option will deny contact from any hosts in its list, without sending a denial response. In the next example, an acl list of mynet is created. Then in the **mytrek.com** zone, only these hosts are allowed to query the server. As the server has no slave DNS serves, zone transfers are disabled entirely. The **blackhole** option denies access from the myrejects list, without sending any rejection notice.

```
acl mynet { 192.168.1.0; };
acl myrejects { 10.0.0.44; 10.0.0.93; };

zone "mytrek.com" {
         type master;
         file "mytrek.com";
         allow-query { mynet; };
         allow-recursion { mynet; };
         allow-transfer { none; };
         blackhole {myrejects};
         };
```

DNSSEC

DNSSEC provides both DNS encrypted authentication. With DNSSEC, you can create a signed zone that is securely identified with an encrypted signature. This form of security is used primarily to secure the connections between master and slave DNS servers, so that a master server transfers update records only to authorized slave servers and does so with a secure encrypted communication. Two servers that establish such a secure connection do so using a pair of public and private keys. In effect, you have a parent zone that can securely authenticate child zones, using encrypted transmissions. This involves creating zone keys for each child, and having those keys used by the parent zone to authenticate the child zones.

You generate a zone key using the **dnssec-keygen** command. A zone key will require the name ZONE (-n) and the name of the zone's domain name. The following example creates a zone key for the **mytrek.com** zone.

```
dnssec-keygen -n ZONE mytrek.com.
```

You can further designate an encryption algorithm (**-a**) and key size (**-b**). Use the **-h** option to obtain a listing of the **dnssec-keygen** options. The following example creates a zone key using a 768-bit key and the DSA encryption algorithm:

```
dnssec-keygen -a DSA -b 768 -n ZONE mytrek.com.
```

dnssec-keygen will create public and private keys, each in corresponding files with the suffixes **.private** and **.key**. The private key is used to generate signatures for the zone, and the public key is used to verify the signatures. You add the public key to the DNS configuration file, **named.conf**, using the $INCLUDE statement to include the **.key** file.

In the **named.conf** file, you then use three DNSSEC DNS resource records to implement secure communications for a given zone: KEY, SIG, and NXT. In these records you use the signed keys for the zones you have already generated. The KEY record holds public keys associated with zones, hosts, or users. The SIG record stores digital signatures and expiration dates for a set of resource records. The NXT record is used to determine that a resource record for a domain does not exist. In addition, several utilities let you manage DNS encryption. With the dnskeygen utility, you generated the public and private keys used for encryption. dnssigner signs a zone using the zone's private key, setting up authentication.

To secure a DNS zone with DNSSEC, you first use dnskeygen to create public and private keys for the DNS zone. Then use dnssinger to create an authentication key. In the DNS zone file, you enter a KEY resource record in which you include the public key. The public key will appear as a lengthy string of random characters. For the KEY record, you enter in the domain name followed by the KEY and then the public key.

```
mytrek.com. KEY 0x4101 3 3 (
AvqyXgKk/uguxkJF/hbRpYzxZFG3x8EfNX389l7GX6w7rlLy
BJl4TqvrDvXr84XsShg+OFcUJafNr84U4ER2dg6NrlRAmZA1
jFfV0UpWDWcHBR2jJnvgV9zJB2ULMGJheDHeyztM1KGd2oGk
Aensm74NlfUqKzy/3KZ9KnQmEpj/EEBr48vAsgAT9kMjN+V3
NgAwfoqgS0dwj5OiRJoIR4+cdRt+s32OUKsclAODFZTdtxRn
vXF3qYV0S8oewMbEwh3trXi1c7nDMQC3RmoY8RVGt5U6LMAQ
KITDyHU3VmRJ36vn77QqSzbeUPz8zEnbpik8kHPykJZFkcyj
jZoHT1xkJ1tk )
```

For authentication, you can sign particular resource records for a given domain or host. Enter the domain or host followed by the term "SIG" and then the resource record's signature.

```
mytrek.com. SIG KEY 3 86400 19990321010705 19990218010705 4932 com. (
Am3tWJzEDzfU1xwg7hzkiJ0+8UQaPtlJhUpQx1snKpDUqZxm
igMZEVk= )
```

The NXT record lets you negatively answer queries.

```
mytrek.com. NXT ftp.mytrek.com. A NS SOA MX SIG KEY NXT
```

To set up secure communications between a parent (master) and child (slave) DNS server, the public key then needs to be sent to the parent zone. There, the key can be signed by the parent. As you may have more than one zone key, you create a keyset using the **dnssec-makekeyset** command. This generates a file with the extension **.keyset**, which is then sent to the parent. The parent zone then uses the **dnssec- signkey** command to sign a child's keyset. This generates a file with the prefix **signedkey-**. This is sent back to the child and now contains both the child's keyset and the parent's signatures. Once the child has the **signedkey-** files, the **dnssec- signedzone** command can be used to sign the zone. The **dnssec-signedzone** command will generate a file with the extension **.signed**. This file is then included in the **named.conf** file with the INCLUDE operation. The **trusted-keys** statement needs to list the public key for the parent zone.

Note TSIG (Transmission SIGnatures) also provide secure DNS communications, but they use a shared private key instead of a private/public key pair. They are usually used for communications between two local DNS servers.

Split DNS: Views

BIND 9.x allows you to divide DNS space into internal and external views. This organization into separate views is referred to as split DNS. Such a configuration is helpful to manage a local network that is connected to a larger network, such as the Internet. You internal view would include DNS information on hosts in the local network, whereas an external view would show only that part of the DNS space that is accessible to other networks. DNS views are often used when you have a local network that you want to protect from a larger network such as the Internet. In effect, you protect DNS information for hosts on a local network from a larger external network such as the Internet.

To implement a split DNS space, you need to set up different DNS servers for the internal and external views. The internal DNS servers will hold DNS information about local hosts. The external DNS server maintains connections to the Internet through a gateway and manages DNS information about any local hosts that allow external access such as FTP or Web sites. The gateways and Internet-accessible sites make up the external view of hosts on the network. The internal servers handle all queries to the local hosts or subdomains. Queries to external hosts such as Internet sites are sent to the external servers, which then forward them on to the Internet. Queries sent to those local hosts that operate external servers such as Internet FTP and Web sites are sent to the external DNS servers for processing. Mail sent to local hosts from the Internet are handled first by the external servers, which then forward them on to the internal servers. With a Split DNS configuration, local hosts can access other local hosts, Internet sites, and local hosts maintaining Internet servers. Internet users, on the other hand, can only access those hosts open to the Internet (served by external servers), such as those with Internet servers like FTP and HTTP. Internet users can, however, send mail messages to any of the local hosts, internal and external.

You can also use DNS views to manage connections between a private network that may use only one Internet address to connect its hosts to the Internet. In this case, the internal view holds the private addresses (192.168…) and the external view connects a gateway host with an Internet address to the Internet. This adds another level of security, providing a result similar to IP masquerading (see Chapter 38).

DNS views are configured with the allow statements such as **allow-query** and **allow-transfer**. With these statements you can specify the hosts that a zone can send and receive queries and transfers from. For example, the internal zone could accept queries from other local hosts, but not from local hosts with external access such as Internet servers. The local Internet servers, though, can accept queries from the local hosts. All Internet queries are forwarded to the gateway. In the external configuration, the local Internet servers can accept queries from anywhere. The gateways receive queries from both the local hosts and the local Internet servers.

In the following example, a network of three internal hosts and one external host is set up into a split view. There are two DNS servers: one for the internal network and one for external access, based on the external host. In reality these make up one network, but they are split into two views. The internal view is known as **mygolf.com** and the external as **greatgolf.com**. In each configuration, the internal hosts are designated in an acl list labeled internals, and the external host is designated in an acl list labeled externals. Should you want to designate an entire IP address range as internal, you could simply use the network address, as in 192.168.1.0/24. In the options section, **allow-query**, **allow-recursion**, and **allow-transfers** restrict access within the network.

The following example shows only the configuration entries needed to implement an internal view. In the **mygolf.com** zone, queries and transfers are allowed only among internal hosts. The global **allow-recursion** option allows recursion among internals.

Internal DNS server

```
acl internals { 192.168.1.1; 192.168.1.2; 192.168.1.3; };
acl externals {10.0.0.1;};
options {
             forward only;
             forwarders {10.0.0.1;}; // forward to external servers
             allow-transfer { none; }; // allow-transfer to no one by
default
             allow-query { internals; externals; };// restrict query access
             allow-recursion { internals; }; // restrict recursion to
internals
              }
zone "mygolf.com" {
               type master;
               file "mygolf";
               forwarders { };
               allow-query { internals; };
               allow-transfer { internals; }
               };
```

In the configuration for the external DNS server, the same acl lists are set up for internals and externals. In the options statement, recursion is now allowed for both externals and internals. In the **mygolf.com** zone, queries are allowed from anywhere, and recursion is allowed for externals and internals. Transfers are not allowed at all.

External DNS server

```
acl internals { 192.168.1.1; 192.168.1.2; 192.168.1.3; };
acl externals {10.0.0.1;};
options {
               allow-transfer { none; }; // allow-transfer to no one
               allow-query { internals; externals; };// restrict query
access
```

```
               allow-recursion { internals; externals }; // restrict
recursion
               };

zone "greatgolf.com" {
               type master;
               file "greatgolf";
               allow-query { any; };
               allow-transfer { internals; externals; };
};
```

Chapter 26: Mail Servers: SMTP, POP, and IMAP

Overview

Mail servers provide Internet users with electronic mail services. They have their own TCP/IP protocols such as Simple Mail Transfer Protocol (SMTP), the Post Office Protocol (POP), and the Internet Mail Access Protocol (IMAP). Messages are sent across the Internet through mail servers that service local domains. A *domain* can be seen as a subnet of the larger Internet, with its own server to handle mail messages sent from or received for users on that subnet. When a user mails a message, it is first sent from his or her host system to the mail server. The mail server then sends the message to another mail server on the Internet, the one servicing the subnet on which the recipient user is located. The receiving mail server then sends the message to the recipient's host system.

At each stage, a different type of operation takes place using different agents (programs). A mail user agent (MUA) is a mail client program, such as mail or Elm. With a MUA, a user composes a mail message and sends it. Then, a mail transport agent (MTA) transports the messages over the Internet. MTAs are mail servers that use the Simple Mail Transfer Protocol (SMTP) to send messages across the Internet from one mail server to another, transporting them from one subnet to another. On Linux and Unix systems, the commonly used MTA is Sendmail, a mail server daemon that constantly checks for incoming messages from other mail servers and sends outgoing messages to appropriate servers. Incoming messages received by a mail server are then distributed to a user with mail delivery agents (MDAs). Most Linux systems use procmail as their MDA, taking messages received by the mail server and delivering them to user accounts (see **www.procmail.org** for more information).

Most Linux distributions automatically install and configure Sendmail for you. On starting your system, you can send and receive messages between local users using Sendmail. You can also set up your Linux system to run a POP server. POP servers hold user's mail until they log in to access their messages, instead of having mail sent to their hosts directly.

Messages sent within a single standalone system require a loopback interface. Most Linux distributions do this automatically for you during the installation process. A *loopback interface* enables your system to address itself, allowing it to send and receive mail to and from itself. A loopback interface uses the hostname **localhost** and a special IP address reserved for use by local systems, 127.0.0.1. You can examine your **/etc/hosts** file to see if your loopback interface has been configured as the localhost. You see "127.0.0.1 localhost" listed as the first entry. If, for some reason, no entry exists for "localhost," you may have to create a loopback interface yourself using the **ifconfig** and **route** commands as shown here (**lo** is the term for loopback):

```
ifconfig lo 127.0.0.1
route add -net 127.0.0.0
```

Received Mail: MX Records

As noted in Chapter 17, a mail address consists of a user name and a host address. The host address takes the form of a fully qualified domain name, listing the host name and the domain name, separated by periods. Most usage of a host name, such as FTP connections, translate the hostname into an IP address and use the IP address to locate the host system. Mail messages operate nearly the same way. However, they make use of the Domain Name Service to determine which host to actually send a message to. The host specified in the mail address may not be the host to which delivery should actually be made. Different networks will often specify a mail server to which mail for the hosts in a network should be delivered. For example, mail addressed to the **rabbit.mytrek.com** host may actually be delivered to the **turtle.mytrek.com** host. **turtle.mytrek.com** may be running a POP mail server that users on **rabbit.mytrek.com** could access to read their mail.

Such mail servers are associated with different hosts by mail exchange records, known as MX records, in a network's DNS configuration (see Chapter 25). When mail is received in a network, the network's DNS configuration is first checked for MX records to determine if the mail is to be delivered to a host different from that in the mail message address. For example, the following MX record says that any mail for the **rabbit.mytrek.com** host is to be delivered to the **turtle.mytrek.com** host. **turtle.mytrek.com** is the mail exchanger for **rabbit.mytrek.com**.

```
rabbit.mytrek.com.   IN   MX    0   turtle.mytrek.com.
```

A host could have several mail exchangers, each with a different priority. If one is down, the one with next highest priority will be accessed. Such a design provides for more robust mail delivery, letting a few well-maintained servers handle received mail, instead of each host on its own.

Mail exchange records are also used for mail addresses for which there are no hosts. For example, you could designate virtual hosts or use the domain name as an address. To use a domain name, you would have an MX record with the domain name mapped to a mail server on the network. Mail addressed to the domain name would be sent to the mail server. For example, with the following MX record, mail sent to **mytrek.com** would be delivered to **turtle.mytrek.com**, which would be running a mail server like Sendmail:

```
mytrek.com.   IN   MX     0   turtle.mytrek.com.
```

Mail addressed to **george@mytrek.com** would be sent to **george@turtle.mytrek.com**.

Note MX records are not only used for mail coming in, but also for mail going out. An MX record can specify a mail server to use for relaying mail from a given host out to a larger network.

MX records come into play with certain Sendmail configurations such as masquerading or centralized mail services. MX records are not required. If you have a standalone system or a small network with only a few hosts, you may want mail received directly by different hosts.

Tip To further secure your e-mail transmissions, you can encrypt them using the Secure Sockets Layer (SSL). With the sslwrap service, POP3, IMAP, and SMTP service can be encrypted with SSL. sslwrap requires that you have installed OpenSSL or ssleay. See **www.rickk.com/sslwrap**.

Sendmail

Sendmail operates as a server to both receive and send mail messages. Sendmail listens for any mail messages received from other hosts and addressed to users on the network hosts it serves. At the same time, Sendmail handles messages users are sending out to remote users, determining to what hosts to send them. You can learn more about Sendmail at **www.sendmail.org** and **www.sendmail.net**, including online documentation and current software packages. The Sendmail newsgroup is **comp.mail.sendmail**. You can also obtain a commercial version from **www.sendmail.com**.

The domain name server for your network designates the host that runs the Sendmail server. This is your mail host. Messages are sent to this host, whose Sendmail server then sends the message to the appropriate user and its host. In your domain name server configuration file, the mail host entry is specified with an MX entry. To print the mail queue of messages for future delivery, you can use **mailq** (or **sendmail -v -q**). This runs Sendmail with instructions to print the mail queue.

The Sendmail software package contains several utilities for managing your Sendmail server (see Table 26-1). These include mailq, which displays the queue of outgoing messages; mailstats, which shows statistics on mail server use; hoststat, which provides the stats of remote hosts that have connected with the mail server; and praliases, which prints out the mail aliases listed in the **/etc/aliases** file. Some, like mailq and hoststat, simply invoke Sendmail with certain options. Others, like mailstats and praliases, are separate programs.

Table 26-1: Sendmail Tools

Tool	**Description**
hoststat	Display status of hosts recently in contact with the mail server (Sendmail).
mailq	Display list of outgoing messages (Sendmail).
newaliases	Generates database version of aliases file (Sendmail).
purgestat	Clears status information (Sendmail).
mailstats	Displays mail server statistics.
makemap	Generates database version of table files. With no argument, it regenerates all database files.
praliases	Print the aliases file.
smrsh	A security tool that restricts programs that Sendmail can run to a secure directory.

Sendmail now maintains all configuration and database files in the **/etc/mail** directory. Here you will find the Sendmail macro configuration file, **sendmail.mc**, as well as several database files (see Table 26-2). Many have changed their names with the release of Sendmail 8.10. For example, the help file is now **/etc/mail/helpfile** instead of **/etc/sendmail.ht**. Specialized files provide support for certain features such as access, which lets you control access by different hosts and networks to your mail server. **virtusertable** lets you designate virtual hosts. These files have both a text and database version. The database version ends with the extension **.db** and is the file actually used by Sendmail. You would make your entries in the text version and then effect the changes by generating a corresponding database version. Database versions are generated using the **makemap** command with the hash option and a redirection operation for the text and database file. For example, to deny access to a particular host, you would place the appropriate entry for it in the **/etc/mail/access** file, editing the file using any text word processor. Then, to generate the **/etc/mail/access.db** version of the access file, you would change to the **/etc/mail directory** and use the following command:

```
cd /etc/mail
makemap hash access < access
```

To regenerate all the database files, just use the **make** command in the **/etc/mail** directory:

```
make
```

Table 26-2: Sendmail Files and Directories

File	Description
/etc/sendmail.cf	Sendmail configuration file (on other systems, this is at **/etc/mail/sendmail.cf**).
/etc/mail/sendmail.mc	Sendmail M4 macro configuration file.
/etc/aliases	Sendmail aliases file for mailing lists.
/etc/aliases.db	Sendmail aliases database file generated by the **newaliases** command using the aliases file.
/etc/mail/access	Sendmail access text file. Access control for screening or relaying messages from different hosts, networks, or users. Used to generate the **access.db** file.
/etc/mail/access.db	Sendmail access database file. Generated from the access text file.
/etc/mail/local-host-names	Sendmail local hosts file for multiple hosts using the same mail server (formerly **sendmail.cw**).
/etc/mail/trusted-users	Sendmail trusted users file (formerly **sendmail.ct**).
/etc/mail/error-header	Sendmail error header file (formerly **sendmail.oE**).
/etc/mail/helpfile	Sendmail help file (formerly **sendmail.ht**).
/etc/mail/statistics	Sendmail statistics file (formerly **sendmail.st**).
/etc/mail/virtusertable	Sendmail virtual user table text file-maps user virtual domain addresses, allowing virtual domains to be hosted on one system. Make entries in this file and then use it to generate the **virtusertable.db** file.
/etc/mail/virtusertable.db	Sendmail virtual user table database generated from the **virtusertable** file.

Table 26-2: Sendmail Files and Directories	
File	**Description**
/etc/mail/mailertable	Sendmail mailer table text file, used to override routing for your domains.
/etc/mail/mailertable.db	Sendmail mailer table database file, generated from the **mailertable** file.
/etc/mail/userdb	Sendmail user database file.
/etc/mail/domaintable	Sendmail **domaintable** file, maps a domain name to another domain name.
/etc/mail/domaintable.db	Sendmail **domaintable** database file, generated from the **domaintable** file.
/var/spool/mail	Incoming mail.
/var/spool/mqueue	Outgoing mail.
/var/spool/maillog	Mail log file.

Certain files and directories are used to manage the mail received and sent. On Red Hat, incoming mail is kept in the **/var/spool/mail** directory and outgoing messages are held in the **/var/spool/mqueue** directory, with subdirectories for different users. Monitoring and error messages are logged in the **/var/log/maillog** file.

Note Red Hat still places the Sendmail configuration file, **sendmail.cf** in the **/etc** directory instead of the **/etc/mail** directory.

Note If your mail sever services several hosts, you will need to enter them in the **/etc/mail/local-host-names** file.

Aliases and LDAP

With Sendmail 8.10, Sendmail can now support the Lightweight Directory Access Protocol (LDAP). LDAP enables the use of a separate server to manage Sendmail queries about user mail addresses. Instead of maintaining aliases and **virtualusertable** files on different servers, LDAP support allows Sendmail to simply use one centralized LDAP server to locate recipients. Mail addresses are simply looked up in the LDAP server, instead of having to search several aliases and **virtualusertable** files on different servers. LDAP also provides secure authentication of users, allowing controlled access to mail accounts. The following example enables LDAP support on Sendmail in the **sendmail.mc** file:

```
FEATURE(`ldap_routing')dnl
LDAPROUTE_DOMAIN(`mytrek.com')dnl
```

Alternatively, Sendmail still supports the use of aliases, either for sent or received mail. It checks an aliases database file called **aliases.db** that holds alias names and their associated e-mail addresses. This is often used for administrator mail, where mail may be sent to the system's root user and then redirected to the mail address of the actual system administrator. You can also alias host addresses, enabling you to address hosts on your network using only their aliases. Alias entries are kept in the **/etc/aliases** file. This file consists of one-line alias records associating aliases with user addresses. You can edit this file to add new entries or to change old ones. They are then stored for lookup in the **aliases.db** file using the command **newaliases**, which runs Sendmail with instructions to update the **aliases.db** file.

Aliases allow you to give different names for an e-mail address or collection of e-mail addresses. One of its most useful features is to create a mailing list of users. Mail addresses to an alias will be sent to the user or list of users associated with the alias. An alias entry consists of an alias name terminated by a colon and followed by a user name or comma-separated list of users. For example, to alias **filmcritic** with the user **george@rabbit.mytrek.com**, you would use the following entry:

```
filmcritic:   george@rabbit.mytrek.com
```

To alias **singers** with the local users **aleina** and **larisa**, you would use

```
singers:     aleina, larisa
```

You can also use aliases as the target addresses, in which case they will expand to their respective user addresses. For example, the **performers** alias will expand through the **filmcritic** and **singers** aliases to the users **george@rabbit.mytrek.com**, **aleina**, and **larisa**.

```
performers:     filmcritic, singers
```

Once you have made your entries in the **/etc/mail/aliases** file, you need to generate a database version using the **newaliases** command:

```
newaliases
```

Sendmail Configuration

The main Sendmail configuration file is **sendmail.cf**, located in the **/etc** directory. This file consists of a sometimes lengthy list of mail definitions that set general options, designate MTAs, and define the address rewrite rules. A series of options set features, such as maximum size of mail messages or the name of host files. The MTAs are those mailers through which Sendmail routes messages. The rewrite rules "rewrite" a mail address to route through the appropriate Internet connections to its destination (these rules can be complex). Check the Sendmail HOW-TO and the online documentation for a detailed explanation.

The **sendmail.cf** definitions can be complex and confusing. To simplify the configuration process, Sendmail supports the use of macros you can use to generate the **sendmail.cf** file using the m4 preprocessor (this requires installation of the sendmail-cf package). Macros are placed in the **/etc/mail/sendmail.mc** file. Here, you can use macros to designate the definitions and features you want for Sendmail, and then the macros are used to generate the appropriate definitions and rewrite rules in the **sendmail.cf** file. As part of the Sendmail package, several specialized versions of the **sendmail.mc** file are made available in the **/usr/share/sendmail-cf** directory. These begin with a system name and have the suffix **.mc**. On Red Hat systems, a specialized Red Hat version is already installed as your **/etc/mail/sendmail.mc** file.

Once you configure your **sendmail.mc** file, you use the following command to generate a **sendmail.cf** file (be sure first to back up your original **sendmail.cf** file). You can rename the

sendmail.mc file to reflect the specific configuration. You can have as many different **.mc** files as you want and use them to implement different configurations.

```
m4 sendmail.mc > /etc/sendmail.cf
```

You then will need to restart the Sendmail server to make the configuration effective:

```
service sendmail restart
```

Note You can also perform basic Sendmail configuration using Linuxconf and Webmin.

In the **sendmail.mc** file, you configure different aspects of Sendmail using either a **define** command to set the value of Sendmail variables or a Sendmail macro that has already been defined to set a particular Sendmail feature. For example, to assign the PROCMAIL_PATH variable to the directory **/usr/bin/procmail**, you would use the following:

```
define('PROCMAIL_MAILER_PATH','/usr/bin/procmail')
```

Similarly, if there are variables that you do not want defined, you can remove them with the **undefine** command:

```
undefine('UUCP_RELAY')
```

To specify the type of operating system that your Sendmail server is running on, you would use the OSTYPE Sendmail macro. The following example specifies the Linux operating system:

```
OSTYPE('linux')
```

The MAILER macro specifies the mail delivery agents (MDAs) to be used. You may have more than one. Usually, you will need a mail delivery agent such as procmail for delivering mail to hosts on your network. In addition, Sendmail in effect operates as an MDA to receive messages from hosts in its local network, which it will then send out to the larger network.

```
MAILER(procmail)
MAILER(smtp)
```

Sendmail also supports an extensive number of features that you need to explicitly turn on. You can do this with the Sendmail FEATURE macro. See Table 26-3 for a list of Sendmail features. The following example turns on the redirect feature, which is used to inform a sender that a recipient is now at a different address:

```
FEATURE(redirect)
```

Table 26-3: Sendmail Features

Feature	Description
use_cw_file	Checks for hosts served by the mail server **/etc/mail/local-host-names** file.
use_ct_file	Reads a list of users from the **/etc/trusted-users** file. These users are trusted users that can change the sender name for their messages.
redirect	Rejects all mail addressed to "address.REDIRECT"

Table 26-3: Sendmail Features	
Feature	**Description**
	providing forwarding address is placed in the **/etc/aliases** file.
nouucp	Does nothing special with UUCP addresses.
nocanonify	Don't pass addresses for canonification.
mailertable	Use a mailer table file, **/etc/mail/mailtertable**, to override routing for particular domains.
domaintable	Uses a domain table file, **/etc/mail/domaintable**, to map one domain to another. Useful if you change your domain name.
uucpdomain	Domain feature for UUCP hosts.
always_add_domain	Adds the local host domain to local mail on your system (those for which you would only need a user name).
allmasquerade	Causes recipient addresses to also masquerade as being from the masquerade host.
masquerade_entire_domain	Masquerades all hosts within the domain specified in **MASQUERADE_AS**.
masquerade_envelope	Masquerade envelope sender and recipient along with headers.
virtusertable	For virtual hosts; maps virtual addresses to real addresses.
nullclient	Turns a Sendmail server into a null client, which simply forwards mail messages to a central mail server for processing.
local_lmtp	Uses an LMTP-capable local mailer.
local_procmail	Uses procmail as the local mailer.
smrsh	Uses the Sendmail Restricted SHell (smrsh) for mailing.
promiscuous_relay	Allows you to relay mail, allowing mail to be received from outside your domain and sent on to hosts outside your domain.
relay_entire_domain	Allows any host in your domain to relay mail (default limits this to hosts in the access database).
relay_hosts_only	Checks for relay permission for particular hosts instead of domains.
accept_unqualified_senders	Allows sender e-mail address to be single user names instead of just fully qualified names that include domain names.
accept_unresolvable_domains	Allows Sendmail to accept unresolvable domain names. Useful for those users in a local network blocked by a firewall from the full DNS namespace. By default, Sendmail requires domains in addresses to be resolvable with DNS.
access_db	Accept or reject mail from domains and hosts in the access

Table 26-3: Sendmail Features

Feature	Description
	database.
blacklist_recipients	Blocks mail to certain users, such as those that should never receive mail-like the users nobody, host, and yp.
rbl	Rejects hosts in the Realtime Blackhole List. Managed by MAPS (Mail Abuse Prevention System LLC) and designed to limit transport of unwanted mass e-mail.**maps.vix.com/rbl/**
ldap_routing	Enables LDAP use.

In addition, you can set certain configuration options. These are variables beginning with the prefix "conf" that you can set and assign values to using the **define** command. There are an extensive number of configuration options, most of which you will not need to change. Table 26-4 lists several of the commonly used ones. The following example defines the **confAUTO_REBUILD** configuration option, which will automatically rebuild the aliases database if needed.

```
define(`confAUTO_REBUILD')
```

Table 26-4: Sendmail Configuration Options

Options	Description
confMAILER_NAME	The sender name used for internally generated outgoing messages.
confDOMAIN_NAME	Your domain name. Used only if your system cannot determine your local domain name.
confCF_VERSION	Appended to the configuration version name.
confCW_FILE	File that holds alternate hostnames for server.
confCT_FILE	File that holds trusted users.
confCR_FILE	File that holds relay domains.
confUSERDB_SPEC	Specifies the user database.
confALIAS_WAIT	Time to wait to rebuild aliases database.
confMIN_FREE_BLOCKS	Minimum number of free blocks needed on file system for Sendmail to accept new mail.
confMAX_MESSAGE_SIZE	The maximum size of messages that will be accepted (in bytes).
confDELIVERY_MODE	Default delivery mode.
confAUTO_REBUILD	Automatically rebuilds alias file if needed.
confERROR_MODE	Error message mode.
confERROR_MESSAGE	Error message header/file.
confBIND_OPTS	Default options for DNS resolver.
confLOG_LEVEL	Set the Sendmail log level.
confTO_CONNECT	The timeout waiting for an initial connect to

Table 26-4: Sendmail Configuration Options	
Options	**Description**
	complete.
confME_TOO	Includes sender in group expansions.
confTO_INITIAL	The timeout waiting for a response on the initial connect.
confTO_IDENT	The timeout waiting for a response to an IDENT query.
confTO_QUEUERETURN	The timeout before a message is returned as undeliverable.
confTRY_NULL_MX_LIST	If the server is the best MX for a host and hasn't made other arrangements, try connecting to the host directly.
confDEF_USER_ID	Default user ID.
confDOUBLE_BOUNCE_ADDRESS	If an error occurs when sending an error message, send that "double bounce" error message to this address.
confDONT_PROBE_INTERFACES	If set, Sendmail will *not* insert the names and addresses of any local interfaces into the list of known "equivalent" addresses.
confMAX_RCPTS_PER_MESSAGE	Allow no more than the specified number of recipients in an SMTP envelope.

Certain macros and types of macros need to be placed in the **sendmail.mc** file in a particular sequence as shown here. Notice that MAILER is toward the end and OSTYPE at the beginning. Local macro definitions (**define**) and FEATURE entries follow the OSTYPE and DOMAIN entries.

```
VERSIONID
OSTYPE
DOMAIN
define
FEATURE
local macro definitions
MAILER
LOCAL_RULE_*
LOCAL_RULESETS
```

The local macro and configuration option definitions that affect a particular feature need to be entered before the FEATURE entry. For example, the **redirect** feature uses the aliases file. Any local definition of the aliases file needs to be entered before the **redirect** feature.

```
define('ALIAS_FILE','/etc/aliases')
FEATURE(redirect)
```

You need to be careful how you enter comments into a **sendmail.mc** file. This file is read as a stream of macros, ignoring all white spaces including newlines. There are no special comment characters that are looked for. Instead, you have to simulate comment indicators using the **dnl** or **divert** commands. The **dnl** command instructs that all characters following that **dnl** command up to and including the next newline are to be ignored. If you place a **dnl** command at the beginning of a text line in the **sendmain.mc** file, it has the effect of turning that line into a comment, ignoring everything on that line-including its newline. Even empty lines will require a **dnl** entry to ignore the newline character:

```
dnl you will have to /etc/sendmail.cf by running this the m4
dnl macro config through preprocessor:
dnl
```

Alternatively you can use the **divert** command. The **divert** command will ignore all data until another **divert** command is reached:

```
divert(-1)
 This is the macro config file used to generate
 the /etc/sendmail.cf file. If you modify the file regenerate
 you will have to regenerate /etc/sendmail.cf by running the m4
 macro
divert(0)
```

For Sendmail to work at all, it only requires that the OSTYPE and MAILERS macros, and any needed features and options, be defined. A very simple Sendmail file is shown here.

mysendmail.mc

```
dnl My sendmail.mc file
OSTYPE(`linux')
define(`PROCMAIL_MAILER_PATH',`/usr/bin/procmail')
FEATURE(redirect)
MAILER(procmail)
MAILER(smtp)
```

A **sendmail.mc** file usually contains many more entries, particularly for parameters and features. The default Red Hat **sendmail.mc** file is shown here.

/etc/sendmail.mc

```
divert(-1)
dnl This is the sendmail macro config file. If you make changes to this
file,
dnl you need the sendmail-cf rpm installed and then have to generate a
dnl new /etc/sendmail.cf by running the following command:
dnl
dnl        m4 /etc/mail/sendmail.mc > /etc/sendmail.cf
dnl
include(`/usr/share/sendmail-cf/m4/cf.m4')
VERSIONID(`linux setup for Red Hat Linux')dnl
OSTYPE(`linux')
define(`confDEF_USER_ID',``8:12'')dnl
undefine(`UUCP_RELAY')dnl
undefine(`BITNET_RELAY')dnl
define(`confAUTO_REBUILD')dnl
define(`confTO_CONNECT', `1m')dnl
define(`confTRY_NULL_MX_LIST',true)dnl
define(`confDONT_PROBE_INTERFACES',true)dnl
define(`PROCMAIL_MAILER_PATH',`/usr/bin/procmail')dnl
define(`ALIAS_FILE', `/etc/aliases')dnl
define(`STATUS_FILE', `/var/log/sendmail.st')dnl
define(`UUCP_MAILER_MAX', `2000000')dnl
define(`confUSERDB_SPEC', `/etc/mail/userdb.db')dnl
define(`confPRIVACY_FLAGS', `authwarnings,novrfy,noexpn,restrictqrun')dnl
define(`confAUTH_OPTIONS', `A')dnl
dnl TRUST_AUTH_MECH(`DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
dnl define(`confAUTH_MECHANISMS', `DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
dnl define(`confTO_QUEUEWARN', `4h')dnl
dnl define(`confTO_QUEUERETURN', `5d')dnl
dnl define(`confQUEUE_LA', `12')dnl
dnl define(`confREFUSE_LA', `18')dnl
dnl FEATURE(delay_checks)dnl
FEATURE(`no_default_msa',`dnl')dnl
FEATURE(`smrsh',`/usr/sbin/smrsh')dnl
FEATURE(`mailertable',`hash -o /etc/mail/mailertable')dnl
FEATURE(`virtusertable',`hash -o /etc/mail/virtusertable')dnl
FEATURE(redirect)dnl
FEATURE(always_add_domain)dnl
FEATURE(use_cw_file)dnl
FEATURE(use_ct_file)dnl
FEATURE(local_procmail)dnl
FEATURE(`access_db')dnl
FEATURE(`blacklist_recipients')dnl
EXPOSED_USER(`root')dnl
dnl This changes sendmail to only listen on the loopback device 127.0.0.1
dnl and not on any other network devices. Comment this out if you want
dnl to accept email over the network.
DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
dnl We strongly recommend to comment this one out if you want to protect
dnl yourself from spam. However, the laptop and users on computers that do
dnl not have 24x7 DNS do need this.
FEATURE(`accept_unresolvable_domains')dnl
MAILER(smtp)dnl
MAILER(procmail)dnl
```

Sendmail Masquerading

For a mail server that is relaying messages from localhosts to the Internet, you may want to masquerade the source of the messages. In large networks that have their own mail servers connected to the Internet, Sendmail masquerading can make messages sent by localhosts appear to be sent by the mail server. Their host address will be replaced by the mail server's address. Returned mail can then be sent to the mail server and held in POP or IMAP server mailboxes that can be later accessed by users on the localhosts. Also, entries in the server's virtualusertable could forward mail to corresponding users in localhosts.

Masquerading is often used to masque localhosts with a domain name. Any subdomains can also be masqueraded. This method can be applied to situations where an ISP or your network administrator have assigned your network its own domain name. You can then masque all mail messages as coming from your domain name instead of from particular hosts or from subdomains you may have. For example, if a network's official domain name is **mytrek.com**, then all messages from the hosts in the **mytrek.com** network such as **rabbit.mytrek.com** and

turtle.mytrek.com, could be masqueraded to appear as just coming from **mytrek.com**. Should the **mytrek.com** network have a subnetwork whose domain is **mybeach.com**, then any messages from **mybeach.com** could also be masqueraded as coming from **mytrek.com**.

You could also use masquerading to allow you to use your own Sendmail server to send mail through an ISP that has not given you your own domain. This is the case for many standalone Internet connections where the ISP connects just one host to the Internet, making it part of its own ISP domain. In this case you would masquerade your local domain as that of the ISP's mail domain. Any mail from hosts in **mytrek.com** would be masqueraded as coming from **myisp.com**. The users sending mail would have to correspond to user mail accounts already set up for you by your ISP. Received mail would still be handled by the ISP mail servers. On the other hand, it is just as easy to use the ISP's mail servers for sending mail (provided they are up and running).

Masquerading is turned on with the **MASQUERADE_AS** command. This takes as its argument the name you want to masquerade your mail as. Normally, the name used is just the domain name, without the mail host. In the following example, the mail is masqueraded as simply **mytrek.com**. Mail sent from a localhost like **turtle.mytrek.com** will appear to be sent by just **mytrek.com**:

```
MASQUERADE_AS('mytrek.com')dnl
```

You will also have to specify the hosts and domains on your local network that your sendmail server should masquerade. If you have decided to masquerade all the hosts in your local network, you just need to set the **masquerade_entire_domain** feature, as in:

```
FEATURE('masquerade_entire_domain')dnl
```

If, instead, you want to masquerade particular hosts or your domain has several subdomains that you want masqueraded, you list them in the **MASQUERADE_DOMAIN** entry. You can list either particular hosts or entire domains. For example, given a local network with the localhosts **turtle.mytrek.com** and **rabbit.mytrek.com** you can list them with the **MASQUERADE_DOMAIN** to have them masqueraded. The domain they are masqueraded as is specified in the MASQUERADE_AS entry.

```
MASQUERADE_DOMAIN('turtle.mytrek.com rabbit.mytrek.com')dnl
```

If you want to masquerade all the hosts in your local network, you can simply list your local network's domain name. Should your local network also support several subdomains, you can list those as well to masquerade them. For example, to masquerade all the hosts in the **mybeach.com** domain, you would use the following entry.

```
MASQUERADE_DOMAIN('mytrek.com mybeach.com')dnl
```

If you have a long list of domains or hosts, or you want to be able to easily change those that should be masqueraded, you can place them in a file to be read by Sendmail. Specify the file with the **MASQUERADE_DOMAIN_FILE** command:

```
MASQUERADE_DOMAIN_FILE('mydomains')dnl
```

If you just want to masquerade all the hosts in your local domain, you use the **masquerade_entire_domain** feature:

```
FEATURE(masquerade_entire_domain)dnl
```

A common configuration for a local network would specify the domain name in the MASQUERADE_AS entry and in the MASQUERADE_DOMAIN entry. Using the example **myisp.com** for the domain, the entries would look like this:

```
MASQUERADE_AS('mytrek.com')dnl
FEATURE(masquerade_entire_domain)dnl
```

If you wanted to masquerade as an ISP's mail domain, you would use the ISP's domain in the MASQUERADE_AS entry as shown here.

```
MASQUERADE_AS('myisp.com')dnl
MASQUERADE_DOMAIN('mytrek.com')dnl
```

You can use the **EXPOSED_USER** feature to override masquerading for certain users. The following example exposes the user **root** in **turtle.mytrek.com**, allowing mail to be addressed using the **turtle** hosts, as in **admin@turtle.mytrek.com**, instead of **admin@mytrek.com**:

```
EXPOSED_USER('root)dnl
```

When mail is received from the outside bearing just the address **mytrek.com**, your network needs to know what host to send it to. This is the host designated as the mail server for the **mytrek.com** network. This information is provided by a mail exchange record (MX) in your DNS configuration that will specify that mail sent to **mytrek.com** will be handled by the mail server-in this case, **turtle.mytrek.com**:

```
mytrek.com.      IN     MX      0      turtle.mytrek.com.
```

You further have to be sure that MX relaying is enabled with the **relay_based_on_MX** feature:

```
FEATURE(relay_based_on_MX)dnl
```

All messages will appear to originate from the mail server's host. For example, if your Sendmail mail server is running on **turtle.mytrek.com**, then mail sent from a localhost called **rabbit.mytrek.com** will appear to have been sent from **turtle.mytrek.com**.

To further masquerade envelopes as well as headers, you add the **masquerade_envelope** feature:

```
FEATURE(masquerade_envelope)dnl
```

You can also masquerade recipient addresses, so that mail sent to users on your localhost will be sent instead to the masqueraded address. Use the **allmasquerade** feature to enable recipient masquerading:

```
FEATURE(allmasquerade)dnl
```

Configuring Mail Servers and Mail Clients

Sendmail can be used as either a mail server, handling mail for various hosts on a network, or as a mail client, managing mail for local users on a particular host. In a simple network configuration, you would have each host running Sendmail in a client configuration, and one host operating as a mail server, relaying mail for the network hosts. For a local network connected to the Internet, your localhosts would run Sendmail in a client configuration, and your gateway would run Sendmail in a server configuration (though the mail server would not have to necessarily run on the gateway). The mail server would relay messages from the local network hosts out to the Internet. The mail server could also be used to block unwanted access from outside hosts, such as those sending spam mail. A basic client or server Sendmail configuration involves just a few features in the **/etc/mail/sendmail.mc** file. The default Red Hat configuration listed in the previous section only allows use on a single host, managing messages between users on that host. To enable client and server use, you will need to make changes to the **/etc/mail/sendmail.mc** file.

Using Sendmail for a Local Network

Red Hat initially configures Sendmail to work only on the system it is running on, localhost. To use Sendmail to send messages to other hosts on a local network, you need to change and add settings in the **sendmail.mc** and **/etc/mail/access** files. A simple network configuration would have Sendmail running on each host, handling both mail sent between users on that host and to send and receive mail to and from users on other hosts. For each Sendmail server configuration, you would make the changes described in Chapter 7.

Alternatively, you could set up a central mail server to handle all the mail on your network. Mail clients on various hosts could send their messages to the central mail server which would then relay them out to the larger network or Internet. Mail could then be received at the central mail server, where clients could later retrieve it. There are several ways to set up a central mail server. One of the simplest is to run a central mail server on your gateway host, and then have nullclient versions of the Sendmail server running on localhosts. Any mail sent from localhosts would be automatically forwarded to the central mail server. Received mail could only be delivered to the central server, usually to a POP or IMAP server also running on the central server's host. Users could then access the POP server to retrieve their mail.

For a centralized configuration, it would make sense to treat users as having their network domain as their address, rather than separate hosts in their network. So the user cece on **rabbit.mytrek.com** would have the mail address **cece@mytrek.com**, not **cece@rabbit.mytrek.com**. Users could have the same name as those on their respective hosts, but corresponding users would be set up on the gateway host to handle received mail managed by the POP or IMAP servers.

An effective simple mail server would involve several components:

- A central mail server running on the gateway host
- Each client running Sendmail as a nullclient
- Masquerade all mail to use the domain address only, not host addresses
- A POP or IMAP server running on the gateway host to handle received mail

Sendmail nullclient Configuration

The nullclient version of Sendmail is a stripped down configuration that simply forwards all mail to the central server. It will not relay mail, nor will it deliver any mail locally. To configure a Sendmail client, you first need to comment out the **DAEMON_OPTIONS** line in the default Red Hat **sendmail.mc** file by placing a **dnl** word in front of it, as shown here. Removing this feature will allow you to receive messages over your local network. This entry is restricting Sendmail to the localhost (127.0.0.1):

```
dnl DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
```

In your network, you will want your Sendmail clients to relay their messages through a central mail server, not to operate as servers themselves. You do this by specifying that they are null clients, as well as listing the mail server operating as the mail hub for their network. To configure Sendmail as a null client, you use the **nullclient** feature. Then define the **MAIL_HUB** feature to list the mail server for the network. In this example, the Sendmail mail server is running on **turtle.mytrek.com**.

```
FEATURE('nullclient')dnl
define('MAIL_HUB', 'turtle.mytrek.com')dnl
```

You could also specify the hub with the **nullclient** feature.

```
FEATURE('nullclient', 'turtle.mytrek.com')dnl
```

Once you have made your changes on a host, restart Sendmail on it:

```
service sendmail restart
```

Sendmail Server Configuration

To configure Sendmail as a server, you need to allow it to accept and relay messages for hosts in your local domain. You do this by adding the feature **relay_entire_domain**:

```
FEATURE(relay_entire_domain)dnl
```

Should your local network also specify mail exchange servers where mail is to be sent and received for certain hosts, then you also have to add the **relay_based_on_MX** feature:

```
FEATURE(relay_based_on_MX)dnl
```

This feature is needed should your DNS configuration include any MX entries specifying mail exchange servers for different hosts or subnetworks.

Be sure to also comment out the **DAEMON_OPTIONS** line as you did for the client configuration:

```
dnl DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
```

The Red Hat default configuration enables the **access_db** feature, which restricts access only to those hosts or networks listed in the **/etc/mail/access** file. The **relay_entire_domain** feature will override the list of hosts in the access file and allow all hosts on your local network to access the mail server. However, should you want more refined control, you could use the access database instead of the **relay_entire_domain** feature to limit access to certain hosts or subdomains.

These changes are enough to configure a mail server. However, if you also want to masquerad your mail, you will have to add masquerading options in the server's **sendmail.mc** file. These are similar to those described earlier in the Sendmail masquerading section. To masquerade all outgoing mail, giving it your domain address, you use the MASQUERADE_AS entry and the **masquerade_entire_domain** feature:

```
FEATURE(masquerade_entire_domain)dnl
MASQUERADE_AS('mytrek.com')dnl
```

In addition, your mail server host has to be configured to accept mail whose messages have just that domain address (not a host address). In this example, the host running the mail server is **turtle.mytrek.com**. This host needs to also be configured to accept mail meant for **mytrek.com**, not just **turtle.mytrek.com**. You can do this by adding **mytrek.com** to the **/etc/mail/local-host-names** file. This makes **mytrek.com**an alias for **turtle.mytrek.com**. Once you have made all your changes, be sure torestart Sendmail:

```
service sendmail restart
```

Receiving Centralized Mail

In a centralized setup, one of the hosts operates as a mail hub in which all mail is delivered to the mail server on that hub. A POP server could also be running on that hub that users could use to access their mail. MX records in the local network's DNS configuration would direct all mail meant for different hosts to the mail hub.

Masquerading would establish a single domain name for all the users on your network, regardless of their hosts. Mail could be sent to just the mail hub and users would access their mail through a POP server. An MX record would direct mail for the masquerade domain to the mail hub.

For example, as noted previously, all users on the **mytrek.com** network would use an address with the domain name **mytrek.com**, as in **chris@mytrek.com**. The hostname would be left out. Masquerading would masquerade any mail sent from **chris@rabbit.mytrek.com** as **chris@mytrek.com**. Received mail would be addressed to **chris@mytrek.com**, which would be directed to an MX mail server, a mail hub, **turtle.mytrek.com**. The MX record would look like this:

```
mytrek.com.       IN    MX   0     turtle.mytrek.com.
```

A corresponding account for each user on all the hosts through the network would be set up at **turtle.mytrek.com**. A POP server on that mail hub could then be used by users to access their mail. With this kind of configuration, e-mail can become network- based instead of host-based. In effect, mail appears to be sent and received directly by all users on the network, instead of through their respective hosts.

Configuring a Workstation with Direct ISP Connection

If you are running a Linux system that is not part of a network, but does have a direct connection to the Internet through an ISP (Internet service provider), then you could simply use the ISP mail servers for sending and receiving mail. Normally, you would have an SMTP mail server for outgoing mail and a POP server for incoming mail. However, you could also configure Sendmail to interface with your ISP.

Be sure to first comment out the **DAEMON_ONLY** option as shown in the previous sections.

Normally, your ISP will provide a mail server that will handle mail for its hosts. To make use of the ISP mail server, you can define it with the **SMART_HOST** option. Mail will be sent through the ISP mail server. **SMART_HOST** has the format type:hostname, where the type is the kind of mail server used, usually SMTP. The default is relay. Define the **SMART_HOST** option to use your ISP to send and receive mail:

```
define ('SMART_HOST', 'smtp:mail.my-isp.com')dnl
```

The **SMART_HOST** option is used to indicate a specific remote mail server that you want to have handle the relaying of your network messages. It can be an ISP mail server, as well as any mail server in a larger network.

For a dial-up connection over a modem, you can use various configuration options to control your connection. The **confMESSAGE_TIMEOUT** option lets you control how long mail can remain on the output queue, letting you keep mail until you are ready to dial in and send it. Setting the **confDELIVERY_MODE** option to **queueonly** lets you send mail when you are ready.

The Mailer Table

The mailer table lets you route messages addressed to a specified host or domain to a particular mail server. You can use the mailer table to have mail addressed to a virtual domain routed to the mail server for your network. To reference an entire domain, prefix the domain name with a period. The host to which the mail is routed is prefixed by the mailer used, usually **smtp** for Sendmail. The following entry will route mail addressed to **.mybeach.com** to the mail server **turtle.mytrek.com**:

```
.mybeach.com        smtp:turtle.mytrek.com
```

Entries are placed in the **/etc/mail/mailertable** file. Once you have made your entries, generate the **mailertable.db** database file with the **make** command:

```
make mailertable
```

Virtual Hosting: Virtual User Table

As seen in Chapter 25, you can define virtual domains for your network. These virtual domains are mapped to one or more real domains by your DNS server. However, you could receive messages with mail addresses for users on your virtual domains. In this case, you would need to map these addresses to users on your real domain so that the mail could be delivered to an existing location. This mapping is carried out by the virtual user table called **/etc/mail/virtusertable**. The virtual user table lets you map mail addresses for virtual domains to users on real domains.

Within a virtual domain, you will have virtual hosts, which will operate off of real hosts on your network. For example, you may have two DNS names for the same host, one real and one virtual. Then, your host operating as your mail server may also have a virtual hostname by which it can be referenced. Such alternate names should be entered in the **local-hostname** file. A mail server supporting a virtual domain will have an alternate name consisting of the virtual domain name.

Virtual domains are designated in your DNS configuration, as discussed in Chapter 25. For such a virtual domain, you will need an MX record to specify which real mail server will handle mail for that virtual domain. Make sure there is an MX record connecting your mail server with your virtual host in the DNS zone file for your network. In the following example, the virtual domain **mybeach.com** will have its mail handled by the mail server **turtle.mytrek.com**:

```
mybeach.com.   IN   MX   10   turtle.mytrek.com.
```

Once your mail server for the virtual domain and any alternate names the server may have are specified, you then need to configure your virtual user table. The virtual user table file is specified in the **sendmail.mc** configuration file with the following entry:

```
FEATURE(`virtusertable', `dbm /etc/mail/virtusertable')dnl
```

In the **/etc/mail** directory you will find two versions of the **virtusertable** file, **virtusertable** and **virtusertable.db**. The **virtusertable** file is an editable text version where you enter your virtual host information. The **virtusertable.db** file is a database version of the text file that can be read by Sendmail. You first edit the **virtusertable** file, making your entries. Then you use it to generate the **virtusertable.db** version that can be used by Sendmail:

```
make virtusertable
```

An entry in the **virtusertable** file consists of a virtual address followed by the real address it is mapped to. In the following example, the users **dylan** and **christopher** at the virtual domain **mybeach.com** are mapped to their actual users on real hosts. **dylan@mybeach.com** is mapped to **dylan@turtle.mytrek.com**, and **christopher@mybeach.com** is mapped to **chris@rabbit.mytrek.com**:

```
dylan@mybeach.com          dylan@turtle.mytrek.com
christopher@mybeach.com    chris@rabbit.mytrek.com
```

You can also map all the addresses for a virtual domain to one real user, as shown here. All the mail to users with the virtual address **mytrains.com** will be sent to the user **trainmail@turtle.mytrek.com**:

```
@mytrains.com              trainmail@turtle.mytrek.com
```

Once you have made your entries, build a **virtusertable.db** database file with the **make** command and the **virtusertable** option, as shown here:

```
make virtusertable
```

Security

Sendmail provides you with the capability of screening out messages from specific domain, host, IP, and user addresses. Rules to perform such screening are kept in the **/etc/mail/access** file. You can edit this file and add your own rules. A rule consists of an address followed by an action to take. The actions supported are listed in Table 26-5. For example, to remove all messages from the **myanoyingad.com** domain, you would enter

```
spamjunk.com DISCARD
```

Table 26-5: Access Actions

Action	**Description**
OK	Accept message even if other rules would reject (exception to the rules).
DISCARD	Discard the message completely.
REJECT	Reject the message, sending a rejection notice to the sender.
RELAY	Relay messages for specified domain.
SMTP-code message	Code and message to be sent to sender.

The next example rejects any message from **larisa@turtle.mycar.com** and sends a notice of the rejection:

```
larisa@turtle.mycar.com REJECT
```

You can also specify an error message to return, as shown here:

```
cecelia@rabbit.mytrek.com    ERROR:"Retired yesterday"
```

To send an error message to spammers, you could include a message as shown here. The first number is an error code.

```
cyberspammer.com     ERROR:"550 We don't accept mail from spammers"
```

An **/etc/mail/access** file with the previous entries would look like the following:

```
spamjunk.com                DISCARD
larisa@turtle.mycar.com     REJECT
cecelia@rabbit.mytrek.com   ERROR:"Retired yesterday"
cyberspammer.com            ERROR:"550 Mail from spammers not accepted"
```

Sendmail actually reads the access rules from a database file called **access.db**, also located in the **/etc/mail** directory. To implement your rules, you have to regenerate the **access.db** file using the access file. You can do this with the **make** command using **access** as the argument, as shown here:

```
make access
```

Sendmail then has to be restarted to read the new **access.db** file.

The use of the access file is enabled in the **sendmail.mc** file with the **access_db** feature:

```
FEATURE('access_db')dnl
```

The access file will deny mail received from the listed addresses. However, you can also reject any mail sent to them. You can also reject received mail for certain hosts on your network. You do this by enabling the **blacklist_recipients** option in the **sendmail.mc** file. This option governs recipients, whereas **access** normally governs senders. Those addresses listed will not be able to receive any mail. This feature is also used for certain administrative users that should never receive mail such as nobody (the guest user) or ftp (the FTP user):

```
FEATURE('blacklist_recipients')dnl
```

The following example will not allow mail to be sent to **cyberspammer.com** (a recipient), nor can mail be received for **justin@lizard.mytrek.com**, **secretproject@rabbit.mytrek.com**, or **mysurfboard.com**:

```
mysurfboard.com                ERROR:"Domain does not exist"
justin@lizard.mytrek.com       "Moved to Hawaii"
secretproject@rabbit.mytrek.com  REJECT
cyberspammer.com               REJECT
```

The Red Hat version of **smb.conf** configures Sendmail to user **access_db**. Access is granted only to users on the localhost. If your system is being used as a mail server for a network, and you have not enabled the **relay_entire_domain** feature, you will need to allow access by other hosts on your network. In the access file, you can place a **RELAY** rule for your network. The **RELAY** rule will let other hosts use your mail server to send messages out to other hosts. This is normally done for a gateway host that needs to relay messages from a local network out to the Internet. The following example allows access from the **mytrekl.com** network:

```
mytrek.com      RELAY
```

For a specific host, place an entry for it in the access file as shown here:

```
rabbit.mytrek.com    RELAY
```

A public list of known spammers is maintained in the Realtime Blackhole List (RBL) by the Mail Abuse Prevention System (MAPS project). Sendmail will automatically check this list for spammers if you use the **rbl** option:

```
FEATURE('rbl')dnl
```

To further secure Sendmail, you should disable the use of **VRFY**. This option allows remote users to try to verify the existence of a user address. This can be used to guess valid users on your system. This option is disabled with the **noverify** feature:

```
FEATURE('noverify')dnl
```

Another potential security breach is the **EXPN** option, which expands mailing lists and aliases to their actual addresses. Use the **noexpn** feature to turn it off:

```
FEATURE('noexpn')dnl
```

By default, Sendmail will refuse mail from any domain that cannot be resolved. You can override this restriction with the **accept_unresolvable_domains** feature. Sendmail will also reject mail whose addresses do not have fully qualified domain names, host and domain. You can override this feature with **accept_unqualified_senders**.

Sendmail Configuration Operators: sendmail.cf

It is not advisable that you modify the **sendmail.cf** file directly. However, it may be helpful to know how it is set up. This section describes the type of entries you will find there. Table 26-6 lists the basic Sendmail configuration operators found in the **sendmail.cf** file. These operators consist of a single uppercase character, some with no spaces separating their arguments. The **D** operator defines macros. These are often used for specific information, such as the name of a host. The macro name usually consists of one character. Uppercase macro names are reserved for use by Sendmail, whereas lowercase macro names are used for user-defined macros. The following example defines a macro called **T** for **turtle.mytrek.com.** You can then reference the macro anywhere in other operations by preceding it with a **$**, as in **$T**. To have a macro name longer than one character, encase the name within braces, as shown here for **rabbit**. To evaluate the **rabbit** macro use **${rabbit}**:

```
DTturtle.mytrek.com
D{rabbit}rabbit.mytrek.com
```

Table 26-6: Sendmail Configuration Operators

Operators	**Action**
D	Define a macro.
C	Define a class.
F	Define a class read from a file.
H	Define mail header.
O	Set an option.
P	Set message priority.
V	Specify version level of **sendmail.cf** file.
K	Specify key file.
M	Specify mailer.
S	Label and start a ruleset.
R	Define a rule.
#	Comment.

Much of the **sendmail.cf** file consists of options that are specified by the **O** operator followed by the option and its arguments. For example, the following entries determine the location of the alias file and the maximum size of a message:

```
O AliasFile=/etc/aliases
O HelpFile=/usr/lib/sendmail.hf
O MaxMessageSize=1000000
```

The **H** operator is used to define mail headers. The **P** operator is used to define the priority of mail messages based on keywords, such as the term "bulk" in the mail header. The **K** operator specifies the location of key database files, such as the **aliases** databases. The **C** operator is used to define a class, such as a collection of hosts, while the **F** operator is used to define a class from names read from a file:

```
# file containing names of hosts for which we receive email
Fw/etc/sendmail.cw
```

The **M** operator is used to define the mailers used by Sendmail. The **S** and **R** operators are used to define rulesets and rewriting rules. Rulesets and rewriting rules are used to determine how a message is to be routed and, if necessary, to rewrite its address so Sendmail's MTAs can handle it. You can think of *rulesets* as functions in a program, which are called as needed to work on message addresses-and can themselves call yet other rulesets. A ruleset consists of a set of rules, much like a function consists of a set of programming statements. Each ruleset is labeled with a number defined by an initial **S** operator. The rules making up the ruleset are then defined by **R** operators. Sendmail uses the rulesets first to format an address into a standard form. Then, for messages being sent, it determines the MTA to use. Special rules called *rewriting rules* can rewrite the address into a form that can be better handled by the MTA. Rewriting rules consist of a left-hand and a right-hand pattern. An address that matches the pattern on the left-hand side is rewritten in the format of the pattern on the right-hand side.

POP Servers

The Post Office Protocol (POP) allows a remote server to hold mail for users who can then fetch their mail from it when they are ready. Unlike Sendmail and procmail, which deliver mail messages directly to a user account on a Linux system, the POP protocol holds mail until a user accesses his or her account on the POP server. The POP server then transfers any received messages to the user's local mailbox. Servers are often used by ISPs to provide Internet mail services for users. Instead of sending mail directly to a user's machine, the mail resides in the POP server until it's retrieved.

You can access the POP server from different hosts; however, when you do, all the messages are transferred to that host. They are not kept on the POP server (though you can set an option to keep them). The POP server simply forwards your messages on the requesting host. When you access your messages from a certain computer, they will be transferred to that computer and erased from the POP server. If you access your POP server again from a different computer, those previous messages will be gone.

The POP protocol provides a set of commands you can use to directly test a POP server. You can access the POP server on port 110 using telnet and then enter a series of POP commands to check the server's performance. The following example connects to the **turtle.mytrek.com** POP server using telnet on port 110:

```
telnet turtle.mytrek.com 110
```

You can then log in to an account using the **USER** and **PASSWORD** commands. The **LIST** command will list messages and the **RETR** command will display a message. Use **DELE** to delete a message. Use the **QUIT** command to end the session.

Note The current version of the POP protocol is known as POP3, whereas POP2 is an earlier one that still may be in use in some places.

Washington POP Server

Red Hat Linux includes in its distribution the University of Washington POP server (ftp.cac.washington.edu/imap), which is part of the University of Washington's **imap** RPM package. Simply install the package, which is already done as part of the standard install (both POP2 and POP3 servers are installed). The server daemons are called **ipop2d** and **ipop3d**. Your Linux system then runs as a POP2 and POP3 server for your network. These servers are run through xinetd. The POP3 server uses the **ipop3** file in the **/etc/xinetd.d** shown here:

```
# default: off
# description: The POP3 service allows remote users to access their mail \
#   using POP3 client such as Netscape Communicator, Mutt, \
#   or fetchmail.
service pop3
{
     disable          = no
     socket_type      = stream
     wait             = no
     user             = root
     server           = /usr/sbin/ipop3d
     log_on_success   += USERID
     log_on_failure   += USERID
}
```

The following command would turn the server on.

```
chkconfig ipop3 on
```

Once you have installed a POP server, you add accounts to it by simply adding standard user accounts on the host it is running on. You would not need to set up a home directory for them, though. Users then access their account using a user name and password set up on the POP server's host. For example, to set up a POP user account for a POP server running on the host **turtle.mytrek.com**, you log in as root on **turtle.mytrek.com** and create a new user as you normally would for that host. To create a POP user account for larisa, just create a larisa user on the **turtle.mytrek.com** host.

Both the POP and IMAP Washington servers support Open SSL authentication and encryption. The SSL enabled versions of the POP and IMAP servers have the names **pop3s** and **imaps**. Use **chkconfig** to turn them on or off.

```
chkconfig pop3s on
```

You also have to have appropriate SSL POP and IMAP certificates installed in the **ssl/certs** directory under the names **ipap3d.pem** and **imapd.pem**.

Qpopper

Qpopper is the current version of the Berkeley POP server (popper). Qpopper is supported by Qualcomm, makers of Eudora e-mail software. The Qpopper Web page is www.eudora.com/free/qpop.html. You can obtain a current source code version from

ftp.qualcomm.com/eudora/servers/unix. RPM package versions are located at distribution sites such as **ftp.redhat.com** for Red Hat and are included in the Red Hat distribution.

Once you have installed Qpopper and have the POP server running, you can add user accounts. Then users on remote systems can access the POP server using various mail clients or fetchmail. fetchmail, as described in Chapter 17, will fetch mail from a user account and place it in the mailbox on their localhost. The following example fetches mail from a POP server running on **turtle.mytrek.com**:

```
fetchmail -p POP3 -u chris   turtle.mytrek.com
```

You can install Qpopper software on your Linux system and have it operate as a POP server for your network. It consists of both the **qpopper** daemon and the popauth program, which manages an authentication database with password encryption for secure user access. popauth creates a database file called **/etc/pop.auth**. To add a user, enter the **popauth** command with the options **-user** and **user name**. You are then prompted for a password with which the user can access his or her POP account.

If you download the source code version, you use the **./configure**, **make**, and **make install** sequence to configure, compile, and install the server. For the configure stage, you may need to enter several options to make Qpopper compatible with your system. For example, if your system uses shadow passwords (as most do), you will need to use the **-enable-specialauth** option. The **-enable-apop** option enables the use of the APOP (Authenticated POP) extension to provide encryption that is managed by the **popauth** command:

```
./configure -- enable-specialauth -enable-apop
```

Qpopper is usually run through xinetd. You would create a popper file in the **/etc/xinetd.d** directory similar to that shown here:

```
service pop-3
{
    socket_type         = stream
    wait                = no
    user                = root
    server              = /usr/bin/popper
    disable             = no
}
```

You can turn the server on or off with the **chkconfig** command:

```
chkconfig pop-3 on
```

Be sure that the pop-3 service is listed in **/etc/services** along with its port, 110:

```
pop-3 110/tcp
```

IMAP

The Internet Mail Access Protocol (IMAP) allows a remote server to hold mail for users who can then log in to access their mail. Unlike the POP servers, IMAP servers retain user mail messages. Users can even save their mail on the IMAP mail server. This has the advantage of keeping a user's mail in one centralized location accessible anywhere on the network. Users can log in to the mail server from any host on the network and read, send, and save their mail. This interactive connection requires more connect time than the POP protocol and is best suited to an Ethernet LAN where users are always connected, rather than dial-up access through a modem.

Unlike POP, IMAP allows users to set up multiple folders on their mail server in which they can organize their mail. IMAP also supports the use of shared folders to which several users can access mail on a given topic.

The Washington University IMAP server is available as an RPM software package, beginning with the name "imap." The name of the server daemon is **imapd**. The server is run through xinetd, and has a file called **imap** in the **/etc/xinetd.d** directory. You would then turn it on or off with the **chkconfig** command:

```
# default: off
# description: The IMAP service allows remote\
# users to access their mail using \
#  an IMAP client such as Mutt, Pine, \
#  fetchmail, or Netscape Communicator.
service imap
{
     socket_type        = stream
     wait               = no
     user               = root
     server             = /usr/sbin/imapd
     log_on_success    += DURATION USERID
     log_on_failure    += USERID
     disable            = no
}
```

The Washington IMAP server also supports SSL as described earlier for the POP server. Also IMAP supports both APOP and CRAM-MD5 authentication methods. These methods use MD checksum values rather than plain text passwords to authenticate users. The sever, however, needs to know the plaintext version of passwords. These are stored in the **/etc/cram-md5.pwd** file. Entries consist of a user name followed by the user's password, separated by a tab. The existence of this file will enable CRAM-MD5 authentication procedures.

Chapter 27: News, Proxy, and Search Servers

Overview

News servers provide Internet users with Usenet news services. They have their own TCP/IP protocol, Network News Transfer Protocol (NNTP). In addition, servers exist that provide better access to Internet resources. Proxy servers speed Web access by maintaining current copies of commonly accessed Web pages, speeding access times by eliminating the need to access the original site constantly. They also perform security functions, protecting servers from unauthorized access. The search and indexing server ht:/Dig enables document searches of Web and FTP sites. With it, you can index documents and carry out complex search requests. Table 27-1 lists different news, proxy, and search servers.

News Servers: INN

The InterNetNews (INN) news server accesses Usenet newsfeeds, providing news clients on your network with the full range of newsgroups and their articles. Newsgroup articles are transferred using the NNTP, and servers that support this protocol are known as *NNTP servers*. INN was written by Rich Salz, and is currently maintained and supported by the Internet Software Consortium (ISC). You can download current versions from its Web site at www.isc.org. INN is also included with most Linux distributions, including Red Hat. The documentation directory for INN in **/usr/share/doc** contains extensive samples. The primary program for INN is the **innd** daemon.

Various INN configuration files can be found in **/etc/news,** including **inn.conf, storage.conf, readers.conf,** and **incoming.conf** (see Table 27-2). **inn.conf** sets options for INN, and the **incoming.conf** file holds the hosts from which you receive newsfeeds. Place entries for remote hosts in the **readers.conf** file to allow them access to your news server. Actual newsfeeds are managed in directories in the **/var/spool/news** directory. Here you will find directories like **article** that holds newsgroup articles, **outgoing** for articles being posted by your users to newsgroups, and **overview**, which holds summary information about articles. Correct configuration of INN can be a complex and time-consuming process, so be sure to consult references and online resources, such as the documents. When you change configurations, be sure to restart the INN server. An **innd** script is in the **/etc/rc.d/init.d** directory, which has arguments similar to the Web **httpd** script. You can use **start**, **restart**, and **stop** arguments with the **innd** script to start, restart, and stop the INN server.

Table 27-1: News, Proxy, and Search Servers

Web Site	Description
www.isc.org	InterNetNews news server, INN
www.leafnode.org	Leafnode news server
squid.nlanr.net	Squid proxy server
www.htdig.org	ht://Dig search and indexing server

Tip There is a Man page for each configuration file in INN, providing detailed information on how to configure their features.

On Red Hat, a basic **inn.conf** file is already set up for you with default settings. Several of the initial parameters you will have to set yourself, such as **domain**, which holds the domain name for your server; **pathhost**, in which you specify the name for your newsreader as you want it to appear in the Path header field for news articles you post; and **server**, in which you specify your newsreader's IP or fully qualified domain name address, like **mynews.mytrek.com**. Different Path options have already been set up for you defining the location of different INN directories, such as **patharticles** set to **/var/spool/news** articles that holds your newsgroup articles, and **pathetc** set to **/etc/news** for your configuration files.

Table 27-2: INN Configuration Files

File	Description
inn.conf	General INN configuration file.
incoming.conf	Specify hosts from which newsfeeds are received.

Table 27-2: INN Configuration Files

File	Description
cycbuff.conf	Configures buffers used in cnfs storage format.
storage.conf	Define storage classes. These consist of a storage method and the newsgroups that use it. Storage methods are the storage formats: tradspool, timehash, timecaf, and cnfs. An additional method, trash, throws out the articles.
expire.ctl	Sets the expiration policy for articles on the news server.
readers.conf	Designates hosts whose users can access the news server with newsreaders.
ovdb.conf	Configures ovdb storage method for overviews.
newsfeeds	Defines how your news server feeds articles to other news servers.
moderated	Moderated newsgroups.
cleanfeed.conf	Configures cleanfeed spam blocking utility.
innfeed.conf	Configures newsfeed processes for innfeed.
innreport.conf	Configures innreport utility for generating log-based reports.
buffindexed.conf	Configures overview buffer for buffindexed method.

Storage formats for the vast number of news articles that are often downloaded and accessed is a central concern for a full-scale news server like INN. INN lets you choose among four possible storage formats: tradspool, timehash, timecaf, and cnfs. tadspool is the traditional method where articles are arranged in a simple directory structure according to their newsgroups. This is known to be very time-consuming to access and store. timehash stores articles in directories organized by the time they were received, making it easier to remove outdated articles. timecaf is similar to timehash, but articles received at a given time are placed in the same file, making access much faster. cnfs stores articles into buffer files that have already been set up. When a buffer file becomes full, the older articles are overwritten by new ones as they come in. This is an extremely fast method since no new files are created. There is no need for setting maximum article limits, but there is also no control on how long an article is retained. In the **storage.conf** file, storage formats are assigned as storage methods to different newsgroups.

INN also supports overviews. These are summaries of articles that readers can check, instead of having to download the entire article to see what it is. Overviews have their own storage methods: tradindexed, buffindexed, and ovdb. You specify the one you want to use in the ovmethod feature in **inn.conf**. tradindexed is fast for readers, but difficult for the server to generate. buffindexed is fast for news servers, but slow for readers. ovdb uses Berkeley DB database files and is very fast for both, but uses more disk space. If you choose ovdb, you can set configuration parameters for it in **ovdb.conf**.

Red Hat has already created a **news** user with a newsgroup for use by your INN daemon and has set up the news directories in **/var/spool/news**. INN software also installs **cron** scripts, which are used to update your news server, removing old articles and fetching new ones. These are usually placed in the **/etc/cron.daily** directory, though they may reside anywhere. **inn-cron-expire** removes old articles and **inn-cron-rnews** retrieves new ones. **inn-cron-nntpsend** sends articles posted from your system to other news servers.

INN also includes several support programs to provide maintenance and crash recovery, and perform statistical analysis on server performance and usage. cleanfeed implements spam protection and innreport generates INN reports based on logs. INN also features a very strong filter system for screening unwanted articles.

Leafnode News Server

Leafnode is a NTTP news server designed for small networks that may have slow connections to the Internet. You can obtain the Leafnode software package along with documentation from its Web site at **www.leafnode.org**. You can also download it from software repositories like **freshmeat.net** or **linuxapps.com**. Along with the Leafnode NNTP server, the software package includes several utilities such as Fetchnews, Texpire, and Newsq. Fetchnews retrieves and sends articles to your upstream news servers. Texpire deletes old news, and Newsq shows articles waiting to be sent out.

Leafnode is tailored to very low usage requirements. Only newsgroups that users on your local network have accessed within the last week are actually downloaded. Newsgroups not accessed in over a week are no longer downloaded. Its usage levels are comparable to a news client, accessing only the news that users on your network want. For a small network, this means having the advantages of a news service with few of the disadvantages. Much less disk space is used and less time is needed to download newsfeeds.

Note Slrnpull is a simple single-user version of Leafnode that can be used only with the slrn newsreader. With it, you can use to automatically download articles in specified newsgroups from your network's news server and view them offline (see Chapter 18).

Configuring Leafnode is a relatively simple process. You need to enter news server information in the Leafnode configuration file and make sure that there is a file for Leafnode in the **xinetd.d** directory so that users on your network can access it. The Leafnode configuration file is **/etc/leafnode/config**. You can edit this file directly or use one of several GUI Leafnode configuration utilities such as Keafnode or leafwa. Keafnode is a KDE program that provides a simple dialog box to configure your Leafnode servers, expire messages in selected newsgroups, and control download operations. leafwa provides a Web-page-based interface letting you configure Leafnode using any Web browser.

You can think of Leafnode as more of a news client that provides news server services to a small local network. It assumes that you already have a connection to a larger network such as the Internet and to an NNTP news server on that network. You then configure Leafnode to download newsgroup articles from that NNTP news server. In the Leafnode configuration file, you need to specify the name of that NNTP news server in the server entry, as shown here:

```
server = mynewsever.mynet.com
```

If you have access to other news servers, you can list them in the supplement entry:

```
supplement = myothernewserver.mynet.com
```

If you need a username and password to access the news server, you can list them in username and password entries following the server entry:

```
username = mylogin
password = mypassword
```

In the configuration file, you can also specify a default expiration time for unread newsgroup articles:

```
expire = 20
```

The Leafnode server does not perform the task of downloading articles. For this task, use Fetchnews. Fetchnews will download articles for specified newsgroups from the NNTP news server and send posted articles to it submitted by users on your local network. Similarly, you use the Texpire application to expire articles that have been previously downloaded. You can further automate these operations by scheduling them in a Leafnode crontab file, **leafnode.crontab**.

To allow users on your network to access the Leafnode server set up for them, you place a file for it in the **xinetd.d** directory that would call **/user/local/sbin/leafnode**.

Squid Proxy-Caching Server

Squid is a proxy-caching server for Web clients, designed to speed Internet access and provide security controls for Web servers. It implements a proxy-caching service for Web clients that caches Web pages as users make requests. Copies of Web pages accessed by users are kept in the Squid cache and, as requests are made, Squid checks to see if it has a current copy. If Squid does have a current copy, it returns the copy from its cache instead of querying the original site. In this way, Web browsers can then use the local Squid cache as a proxy HTTP server. Squid currently handles Web pages supporting the HTTP, FTP, Gopher, SSL, and WAIS protocols (Squid cannot be used with FTP clients). Replacement algorithms periodically replace old objects in the cache.

As a proxy, Squid does more that just cache Web objects. It operates as an intermediary between the Web browsers (clients) and the servers they access. Instead of connections being made directly to the server, a client connects to the proxy server. The proxy then relays requests to the Web server. This is useful for situations where a Web server is placed behind a firewall server, protecting it from outside access. The proxy is accessible on the firewall, which can then transfer requests and responses back and forth between the client and the Web server. The design is often used to allow Web servers to operate on protected local networks and still be accessible on the Internet. You can also use a Squid proxy to provide Web access to the Internet by local hosts. Instead of using a gateway providing complete access to the Internet, local hosts could use a proxy to allow them just Web access (see Chapter 7). You could also combine the two, allowing gateway access, but using the proxy server to provide more control for Web access. In addition, the caching capabilities of Squid would provide local hosts with faster Web access.

Technically, you could use a proxy server to simply manage traffic between a Web server and the clients that want to communicate with it without doing caching at all. Squid combines both capabilities as a proxy-caching server.

Squid also provides security capabilities that let you exercise control over hosts accessing your Web server. You can deny access by certain hosts and allow access by others. Squid also supports the use of encrypted protocols such as SSL (see Chapter 39). Encrypted communications are tunneled through (passed through without reading) the Squid server directly to the Web server.

Squid is supported and distributed under a GNU Public License by the National Laboratory for Applied Network Research (NLANR) at the University of California, San Diego. The work is based on the Harvest Project to create a Web indexing system that included a high-performance cache daemon called **cached**. You can obtain current source code versions and online documentation from the Squid home page at **http://squid.nlanr.net** and the Squid FTP site at **ftp.nlanr.net**. The Squid software package consists of the Squid server, a domain name lookup program called dnsserver, an FTP client called ftpget, and a cache manager script called **cachemgr.cgi**. The dnsserver resolves IP addresses from domain names, and the ftpget program is an FTP client Squid uses to retrieve files from FTP servers. **cachemgr.cgi** lets you view statistics for the Squid server as it runs.

On Red Hat, you can start, stop, and restart the squid server using the squid service script, as shown here:

```
service squid restart
```

Configuring Client Browsers

For users on a host to access the Web through a proxy, they need to specify their proxy server in their Web browser configuration. For this you will need the IP address of the host running the Squid proxy server as well as the port it is using. Proxies usually make use of port 3128. To configure use of a proxy server running on the local sample network described in Chapter 7, you would enter the following. The proxy server is running on **turtle.mytrek.com** (192.168.0.1) and using port 3128.

```
192.168.0.1 3128
```

On Mozilla and Netscape, the user on the sample local network would first select the Proxy panel located in Preferences under the Edit menu. Then, in the Manual proxy configuration's View panel, enter the previous information. You will see entries for FTP, Gopher, HTTP, Security, and WAIS proxies. For standard Web access, enter the IP address in the FTP, Gopher, and Web boxes. For their port boxes, enter 3128.

For Konqueror on the KDE Desktop, select the Proxies panel on the Settings window. Here, you can enter the proxy server address and port numbers.

If your local host is using Internet Explorer (such as a Windows system does), you would set the proxy entries in the Local Area Network settings accessible from the Internet Options window.

On Linux or Unix systems, local hosts can set the **http_proxy**, **gopher_proxy**, and **ftp_proxy** shell variables to configure access by Linux-supported Web browsers such as lynx. You can place these definitions in your **.bash_profile** or **/etc/profile** files to have them automatically defined whenever you log in.

```
http_proxy=192.168.0.1:3128
ftp_proxy=192.168.0.1:3128
```

```
gopher_proxy=192.168.0.1:3128
export http_proxy ftp_proxy gopher_proxy
```

Before a client on a local host could use the proxy server, access permission would have to be given to them in the server's **squid.conf** file, described in the following section on security. Access can be provided to an entire network easily. For the sample network used here, you would have to place the following entries in the **squid.conf** file. These are explained in detail in the following sections.

```
acl mylan src 192.168.0.0/255.255.255.0
http_access allow mylan
```

Tip Web clients that need to access your Squid server will need to know the server's address and the port for Squid's HTTP services, by default 3128.

squid.conf

The Squid configuration file is **squid.conf**, located in the **/etc/squid** directory. In the **/etc/squid/squid.conf** file, you set general options like ports used, security options controlling access to the server, and cache options for configuring caching operations. You can use a backup version called **/etc/squid/squid.conf.default** to restore your original defaults. The default version of **squid.conf** provided with Squid software includes detailed explanations of all standard entries, along with commented default entries. Entries consist of tags that specify different attributes. For example, **maximum_object_size** and **maximum_object** set limits on objects transferred.

```
maximum_object_size 4096 KB
```

As a proxy, squid will use certain ports for specific services, such as port 3128 for HTTP services like Web browsers. Default port numbers are already set for squid. Should you need to use other ports, you can set them in the **/etc/squid/squid.conf** file. The following entry shows how you would set the Web browser port:

```
http_port 3128
```

Note Squid uses the Simple Network Management Protocol (SNMP) to provide status information and statistics to SNMP agents managing your network. You can control SNMP with the snmp access and port configurations in the **squid.conf** file.

Security

Squid can use its role as an intermediary between Web clients and a Web server to implement access controls, determining who can access the Web server and how. Squid does this by checking access control lists (ACLs) of hosts and domains that have had controls placed on them. When it finds a Web client from one of those hosts attempting to connect to the Web server, it executes the control. Squid supports a number of controls with which it can deny or allow access to the Web server by the remote host's Web client (see Table 27-3). In effect, Squid sets up a firewall just for the Web server.

Table 27-3: Squid ACL Options

Option	Description
src *ip-address/netmask*	Client's IP address

Table 27-3: Squid ACL Options

Option	**Description**
src *addr1-addr2/netmask*	Range of addresses
dst *ip-address/netmask*	Destination IP address
myip *ip-address/netmask*	Local socket IP address
srcdomain *domain*	Reverse lookup, client IP
dstdomain *domain*	Destination server from URL; for **dstdomain** and **dstdom_regex**, a reverse lookup is tried if an IP-based URL is used
srcdom_regex [-i] *expression*	Regular expression matching client name
dstdom_regex [-i] *expression*	Regular expression matching destination
time *[day-abbrevs] [h1:m1-h2:m2]*	Time as specified by day, hour, and minutes. Day abbreviations: S - Sunday, M - Monday, T - Tuesday, W - Wednesday, H - Thursday, F - Friday, A - Saturday
url_regex [-i] *expression*	Regular expression matching on whole URL
urlpath_regex [-i] *expression*	Regular expression matching on URL path
port *ports*	Specify a port or range of ports
proto *protocol*	Specify a protocol, such as HTTP or FTP
method *method*	Specify methods, such as GET and POST
browser [-i] regexp	Pattern match on user-agent header
ident *username*	String match on **ident** output
src_as *number*	Used for routing of requests to specific caches
dst_as *number*	Used for routing of requests to specific caches
proxy_auth *username*	List of valid usernames
snmp_community *string*	A community string to limit access to your SNMP agent

The first step in configuring Squid security is to create ACLs. These lists are lists of hosts and domains for which you want to set up control. You define ACLs using the **acl** command, in which you create a label for the systems on which you are setting controls. You then use commands, such as **http_access**, to define these controls. You can define a system, or a group of systems, based on several **acl** options, such as the source IP address, the domain name, or even the time and date (refer back to Table 27-2). For example, the **src** option is used to define a system or group of systems with a certain source address. To define a **mylan acl** entry for systems in a local network with the addresses 192.168.0.0 through 192.168.0.255, use the following ACL definition:

```
acl mylan src 192.168.0.0/255.255.255.0
```

Once defined, an ACL definition can be used in a Squid option to specify a control you want to place on those systems. For example, to allow access by the mylan group of local systems to the Web through the proxy, use an **http_access** option with the **allow** action specifying **mylan** as the **acl** definition to use, as shown here:

```
http_access allow mylan
```

By defining ACLs and using them in Squid options, you can tailor your Web site with the kind of security you want. The following example allows access to the Web through the proxy by only the mylan group of local systems, denying access to all others. Two **acl** entries are set up: one for the local system and one for all others. **http_access** options first allow access to the local system, and then deny access to all others.

```
acl mylan src 192.168.0.0/255.255.255.0
acl all src 0.0.0.0/0.0.0.0
http_access allow mylan
http_access deny all
```

The default entries that you will find in your **squid.conf** file, along with an entry for the mylan sample network, are shown here. You will find these entries in the ACCESS CONTROLS section of the **squid.conf** file.

```
acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl mylan src 192.168.0.0/255.255.255.0
acl SSL_ports port 443 563
```

The order of the **http_access** options is important. Squid starts from the first and works its way down, stopping at the first **http_access** option with an ACL entry that matches. In the previous example, local systems that match the first **http_access** command are allowed, whereas others fall through to the second **http_access** command and are denied.

For systems using the proxy, you can also control what sites they can access. For a destination address, you create an **acl** entry with the **dst** qualifier. The **dst** qualifier takes as its argument the site address. Then you can create an **http_access** option to control access to that address. The following example denies access by anyone using the proxy to the destination site **rabbit.mytrek.com**. If you have a local network accessing the Web through the proxy, you can use such commands to restrict access to certain sites.

```
acl myrabbit dst rabbit.mytrek.com
http_access deny myrabbit
```

The **http_access** entries already defined in the **squid.conf** file, along with an entry for the mylan network, are shown here. Access to outside users is denied, whereas access by hosts on the local network and the localhost (squid server host) is allowed.

```
http_access allow localhost
http_access allow mylan
http_access deny all
```

You can also qualify addresses by domain. Often, Web sites can be referenced using only the domain. For example, a site called **www.mybeach.com** can be referenced using just the domain **mybeach.com.** To create an **acl** entry to reference a domain, use either the **dstdomain** or **srcdomain** options for destination and source domains, respectively.

Remember, such a reference refers to all hosts in that domain. An **acl** entry with the **dstdomain** option for **mybeach.com** restricts access to **www.mybeach.com**, **ftp.mybeach.com**, **surf.mybeach.com**, and so on. The following example restrictsaccess to the **www.mybeach.com** site along with all other **.mybeach.com** sites and any hosts in the **mybeach.com** domain:

```
acl thebeach dstdomain .mybeach.com
http_access deny thebeach
```

You can list several domains or addresses in an **acl** entry to reference them as a group, but you cannot have one domain that is a subdomain of another. For example, if **mybeachblanket.com** is a subdomain of **mybeach.com**, you cannot list both in the same **acl** list. The following example restricts access to both **mybeach.com** and **mysurf.com**:

```
acl beaches dstdomain .mybeach.com .mysurf.com
http_access deny beaches
```

An **acl** entry can also use a pattern to specify certain addresses and domains. In the following example, the access is denied to any URL with the pattern "chocolate", and allows access from all others:

```
acl Choc1 url_regex chocolate
http_access deny Choc1
http_access allow all
```

Squid also supports ident and proxy authentication methods to control user access. The following example only allows the users **dylan** and **chris** to use the Squid cache:

```
ident_lookup on
acl goodusers user chris dylan
http_access allow goodusers
http_access deny all
```

Caches

Squid uses the Internet Cache Protocol (ICP) to communicate with other Web caches. Using the ICP protocols, your Squid cache can connect to other Squid caches or other cache servers, such as Microsoft proxy server, Netscape proxy server, and Novell BorderManager. This way, if your network's Squid cache does not have a copy of a requested Web page, it can contact another cache to see if it is there instead of accessing the original site. You can configure Squid to connect to other Squid caches by connecting it to a cache hierarchy. Squid supports a hierarchy of caches denoted by the terms *child*, *sibling*, and *parent*. Sibling and child caches are accessible on the same level and are automatically queried whenever a request cannot be located in your own Squid's cache. If these queries fail, a parent cache is queried, which then searches its own child and sibling caches-or its own parent cache, if needed-and so on. Use **cache_host** to set up parent and sibling hierarchical connections:

```
cache_host sd.cache.nlanr.net parent 3128 3130
```

You can set up a cache hierarchy to connect to the main NLANR server by registering your cache using the following entries in your **squid.conf** file:

```
cache_announce 24
announce_to sd.cache.nlanr.net:3131
```

Squid keeps several logs. **access.log** holds requests sent to your proxy, **cache.log** holds Squid server messages such as errors and startup messages, and **store.log** holds information about the Squid cache such as objects added or removed. You can use the cache manager (**cachemgr.cgi**) to manage the cache and view statistics on the cache manager as it runs. To run the cache manager, use your browser to execute the **cachemgr.cgi** script (this script should be placed in your Web server's **cgi-bin** directory). You can also monitor Squid using the Multi Router Traffic utility.

Dig Server

Dig, known officially as ht:/Dig, is a Web indexing and search system designed for small networks or intranets. Dig is not considered a replacement for full-scale Internet search systems, such as Lycos, Infoseek, or Alta Vista. Unlike Web server-based search engines, Dig can span several Web servers at a site. Dig was developed at San Diego State University and is distributed free under the GNU Public License. You can obtain information and documentation at **www.htdig.org**, and you can download software packages-including RPM packages-from **ftp.htdig.org**.

Dig supports simple and complex searches, including complex Boolean and fuzzy search methods. *Fuzzy searching* supports a number of search algorithms, including exact, soundex, and synonyms. Searches can be carried out on both text and HTML documents. HTML documents can have keywords placed in them for more accurate retrieval, and you can also use HTML templates to control how results are displayed.

Searches can be constrained by authentication requirements, location, and search depth. To protect documents in restricted directories, Dig can be informed to request a specific username and password. You can also restrict a search to retrieve documents in a certain URL, to search subsections of the database, or to retrieve only documents that are a specified number of links away.

All the ht:/Dig programs use the same configuration file, **htdig.conf**, located in the **/etc/htdig** directory. The configuration file consists of attribute entries, each beginning with the attribute line and followed by the value after a colon. Each program takes only the attributes it needs:

```
max_head_length: 10000
```

You can specify attributes such as **allow_virtual_hosts**, which index virtual hosts (see Chapter 25) as separate servers, and **search_algorithm**, which specifies the search algorithms to use for searches.

Dig consist of five programs: htdig**,** htmerge**,** htfuzzy**,** htnotify**,** and htsearch. htdig, htmerge, and htfuzzy generate the index, while htsearch performs the actual searches. First, htdig gathers information on your database, searching all URL connections in your domain and associating Web pages with terms. The htmerge program uses this information to create a searchable database, merging the information from any previously generated database. htfuzzy creates indexes to allow searches using fuzzy algorithms, such as soundex and synonyms. Once the database is created, users can use Web pages that invoke htsearch to search this index. Results are listed on a Web page. You can use META tags in your HTML documents to enter specific htdig keywords, exclude a document from indexing, or provide notification information such as an e-mail address and an expiration date. htnotify uses the e-mail and expiration date to notify Web page authors when their pages are out-of-date.

htsearch is a CGI program that expects to be invoked by an HTML form, and it accepts both the GET and POST methods of passing data. The htsearch program can accept a search request from any form containing the required configuration values. Values include search features such as config (configuration file), method (search method), and sort (sort criteria). For the Web page form that invokes htsearch, you can use the default page provided by htdig or create your own. Output is formatted using templates you can modify. Several sample files are included with the htdig software: **rundig** is a sample script for creating a database, **searchform.html** is a sample HTML document that contains a search form for submitting htdig searches, **header.html** is a sample header for search headers, and **footer.html** is for search footers.

Part VI: System Administration

Chapter List

Chapter 28: Basic System Administration

Overview

Linux is designed to serve many users at the same time, as well as to provide an interface among the users and the computer with its storage media, such as hard disks and tapes. Users have their own shells through which they interact with the operating system, but you may need to configure the operating system itself in different ways. You may need to add new users, printers, and even file systems. Such operations come under the heading of system administration. The person who performs such actions is referred to as either a *system administrator* or a *superuser.* In this sense, two types of interaction with Linux exist: regular users' interaction and the superuser, who performs system administration tasks. The chapters in the "Administration" section cover operations such as changing system runlevels, managing users, and configuring printers and compiling the kernel. You perform most of these tasks, such as adding a new printer or mounting a file system, rarely. Other tasks, such as adding users, you perform on a regular basis. Basic system administration covers topics such as system access by superusers, selecting the run level to start, system configuration files, and performance monitoring. These are discussed in detail in this chapter.

System Management: Superuser

To perform system administration operations, you must first have the correct password that enables you to log in as the root user, making you the superuser. Because a superuser has the power to change almost anything on the system, such a password is usually a carefully guarded secret given only to those whose job is to manage the system. With the correct password, you can log into the system as a system administrator and configure the system in different ways. You can start up and shut down the system, as well as change to a different operating mode, such as a single-user mode. You can also add or remove users, add or remove whole file systems, back up and restore files, and even designate the system's name. To become a superuser, you log into the root user account. This is a special account reserved for system management operations with unrestricted access to all components of your Linux operating system. When you log into the system as the root user, you are placed in a shell from which you can issue administrative Linux commands. The prompt for this shell is a sharp sign, #. In the next example, the user logs into the system as the root user. The password is, of course, not displayed.

```
login: root
password:
#
```

As the root user, you can use the **passwd** command to change the password for the root login, as well as for any other user on the system.

```
# passwd root
New password:
Re-enter new password:
#
```

While you are logged into a regular user account, it may be necessary for you to log into the root and become a superuser. Ordinarily, you would have to log out of your user account first, and then log into the root. Instead, you can use the **su** command (switch user) to log in directly to the root while remaining logged into your user account. A CTRL-D or **exit** command returns you to your own login. When logged in as the root, you can use **su** to log in as any user, without providing the password. In the next example, the user is logged in already. The **su** command then logs the user into the root, making the user a superuser. Some basic superuser commands are shown in Table 28-1.

```
$ pwd
/home/chris
$su
 password:
# cd
# pwd
/root
# exit
$
```

Table 28-1: Basic System Administration

Command	Description
su root	Logs a superuser into the root from a user login; the superuser returns to the original login with a CTRL-D.

Table 28-1: Basic System Administration

Command	Description
passwd *login-name*	Sets a new password for the login name.
crontab *options file-name*	With *file-name* as an argument, installs **crontab** entries in the file to a **crontab** file; these entries are operations executed at specified times: **-e** Edits the **crontab** file **-l** Lists the contents of the **crontab** file **-r** Deletes the **crontab** file
telinit *runelevl*	Changes the system runlevels (see Table 28-2).
lilo *options Config-file*	Reinstalls the Linux Loader (LILO).
shutdown *options time*	Shuts down the system; similar to CTRL-ALT-DEL.
date	Sets the date and time for the system.
TimeTool	GUI tool to set system time and date.
Kcron	KDE cron management tool.

Note For security reasons, Linux distributions do not allow the use of **su** in a telnet session to access the root user.

System Configuration

Although many different specialized components go into making up a system, such as servers, users, and devices, some operations apply to the system in general. These include setting the system date and time, specifying shutdown procedures, and determining the services to start up and run whenever the system boots. In addition, you can use numerous performance analysis tools to control processes and check on resource use.

System Time and Date

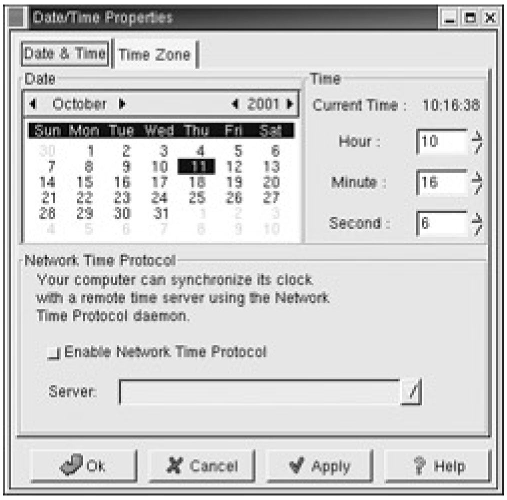
You can use several different tools to set the system time and date, depending on the distribution you use. On all distributions, you can set the system time and date using the shell **date** command. Most users prefer to use a configuration tool. On Red Hat, you can also use the Control Panel Time tool. Recall that you set the time and date when you first installed your system. You should not need to do so again. If you entered the time incorrectly or moved to a different time zone, though, you could use this utility to change your time.

You can use the **date** command on your root user command line to set the date and time for the system. As an argument to **date**, you list (with no delimiters) the month, day, time, and year. In the next example, the date is set to 2:59 P.M., March 6, 2000 (03 for March, 06 for the day, 1459 for the time, and 00 for the year 2000):

```
# date 0306145900
Mon Mar 6 02:59:27 PST 2000
```

You can also use the Red Hat Date and Time Properties utility to change the time, date, and time zone. Select it on the System Settings window accessible from the Start Here window. There are two panels, one for the date and time and one for the time zone. Use the calendar to select the year, month, and date. Then use the Time boxes to set the hour, minute, and second.

The Network Time Protocol is also supported, allowing your date and time to be set by a remote server. The Time Zone panel shows a map with locations. Select one nearest you to set your time zone.



Current Time : 10:16:38

Hour : 10

Minute : 16

Second : 6

Your computer can synchronize its clock with a remote time server using the Network Time Protocol daemon.

☐ Enable Network Time Protocol

Server:

Note You can also set the time and date with the Date & Time tool in the KDE Control Center (system). You can also change the time and date on Webmin or Linuxconf, should you have them installed.

Scheduling Tasks: crontab

Although it is not a system file, a crontab file is helpful in maintaining your system. A *crontab* file lists actions to take at a certain time. The **cron** daemon constantly checks the user's crontab file to see if it is time to take these actions. Any user can set up a crontab file of his or her own. The root user can set up a crontab file to take system administrative actions, such as backing up files at a certain time each week or month.

A crontab entry has six fields: the first five are used to specify the time for an action, while the last field is the action itself. The first field specifies minutes (0-59), the second field specifies the hour (0-23), the third field specifies the day of the month (1-31), the fourth field specifies the month of the year (1-12), and the fifth field specifies the day of the week (0-6), starting with 0 as Sunday. In each of the time fields, you can specify a range, a set of values, or use the asterisk to indicate all values. For example, 1-5 for the day-of-week field specifies Monday through Friday. In the hour field, 8, 12, 17 would specify 8 A.M., 12 noon, and 5 P.M. An ***** in the month-of-year field indicates every month. The following example backs up the **projects** directory at 2:00 A.M. every weekday:

```
0 2 * * 1-5 tar cf /home/chris/backp /home/chris/projects
```

You use the **crontab** command to install your entries into a crontab file. To do this, you first create a text file and type your crontab entries. Save this file with any name you want, such as **mycronfile**. Then, to install these entries, enter **crontab** and the name of the text file. The **crontab** command takes the contents of the text file and creates a crontab file in the **/var/spool/cron** directory, adding the name of the user who issued the command. In the next example, the root user installs the contents of the **mycronfile** as the root's crontab file. This creates a file called **/var/spool/cron/root**. If a user named **justin** installed a crontab file, it would create a file called **/var/spool/cron/justin**. You can control use of the **crontab** command by regular users with the **/etc/cron.allow** file. Only users with their names in this file can create crontab files of their own.

```
# crontab mycronfile
```

Never try to edit your crontab file directly. Instead, use the **crontab** command with the **-e** option. This opens your crontab file in the **/var/spool/cron** directory with the standard text editor, such as Vi. **crontab** uses the default editor as specified by the EDITOR shell environment variable. To use a different editor for **crontab**, change the default editor by assigning the editor's program name to the EDITOR variable and exporting that variable. Running **crontab** with the **-l** option displays the contents of your crontab file, and the **-r** option deletes the entire file. Invoking **crontab** with another text file of crontab entries overwrites your current crontab file, replacing it with the contents of the text file.

To more easily manage and create cron jobs, you can use the Kcron utility, accessible on the KDE desktop. When you use Kron as the root user, it will list all the users on your system, and allow you to create and manage jobs for them (see Figure 28-1). These include the system cron jobs as specified in the system crontab file. For each user you can specify the jobs to be performed as well as any variables you want to set for those jobs. Each user entry has two folders, one for cron jobs, named Tasks, and the other for the jobs' variables, named Variables. You can add entries to either. The Edit menu will list operations you can perform on jobs such as disabling them, running them immediately, modifying their times, or deleting them altogether. When you create or modify a job, the Edit Task dialog box opens up where you can select the months, days, hours, and minutes when to run a job. You also select the program to run along with its arguments. Once you have made your changes, you select Save from the File menu to save them to your crontab file. Individual users can also run Kcron to manage their own cron jobs. In this case, Kcron will only show a single Tasks and Variables folder, where the user can enter his or her jobs and variables.

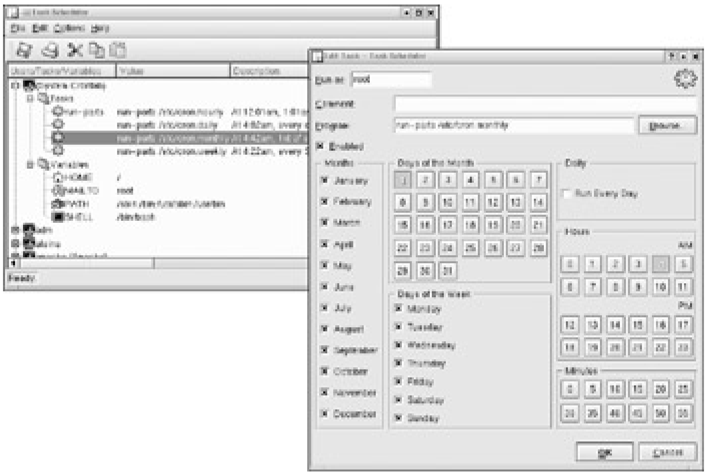


Figure 28-1: Kcron

Note You can also schedule tasks with crontab using Webmin. On Webmin, select the Scheduled Cron Jobs icon on the System panel. This opens a page listing the scheduled jobs in your crontab file where you can create and edit cron jobs.

System Runlevels: telinit and shutdown

Your Linux system can run in different states, depending on the capabilities you want to give it. For example, you can run your system in an administrative state, keeping user access shutdown. Normal full operations are activated by simply running your system at a certain state. These states (also known as *modes*) are referred to as *runlevels*, the level of support at which you are running your system. Your Linux system has several runlevels, numbered from 0 to 6. When you power up your system, you enter the default runlevel. You can then change to other runlevels with the **telinit** command. Runlevels 0, 1, and 6 are special runlevels that perform certain functions. Runlevel 0 is the power-down state and is invoked by the **halt** command. The command **telinit** 0 shuts down your system. Runlevel 6 is the reboot state-it shuts down the system and reboots. Runlevel 1 is the single-user state, allowing access only to the superuser, and does not run any network services. This enables you, as administrator, to perform administrative actions without interference from others. Other runlevels reflect how you want the system to be used. Runlevel 2 is a partial multiuser state, allowing access by many users, but without network services like NFS or xinetd. It is useful for a system that is not part of a network. Both runlevel 3 and runlevel 5 run a fully operational Linux system, with multiuser support and remote file sharing access. They differ in terms of the interface they use. Runlevel 3 starts up your system with the command line interface (also known as the *text mode interface*). Runlevel 5 starts up your system with an X session, running the X Window System server and invoking a graphical login, using display managers, such as gdm or xdm. If you choose to use graphical logins during installation, this will be your default runlevel. The runlevels are listed in Table 28-2.

Table 28-2: System Runlevels (states)

State	Description
telinit *runlevel*	Changes the system runlevel; you can use it to power up or power down a system, or allow multiuser or single-user access; the **telinit** command takes as its argument a number representing a system state. **telinit** is a link to **init** which performs the state change operation.
System Runlevel (state)	
0	Halt (do *not* set the default to this); this shuts down the system completely.
1	Administrative single-user mode; denies other users access to the system, but allows root access to the entire multiuser file system. Startup scripts are not run. (Use **s** or **S** to enter single user mode with startup scripts run).
2	Multiuser, without network services like NFS, xinetd, and NIS (the same as 3, but you do not have networking).
3	Full multiuser mode with login to command line interface; allows remote file sharing with other systems on your network. Also referred to as the text mode state.
4	Unused.
5	Full multiuser mode that starts up in an X session, initiating a graphical login; allows remote file sharing with other systems on your network (same as 3, but with graphical login).
6	Reboots; shuts down and restarts the system (do *not* set the default to this).

Changing runlevels can be helpful should you have problems at a particular runlevel. For example, if your video card is not installed properly, then any attempt to start up in runlevel 5 will likely crash your system because at this level your graphical interface starts immediately. Instead, you would want to use the command line interface, runlevel 3, to first fix your video card installation.

Note You can use the single user runlevel (1) as a recovery mode state, allowing you to start up your system without running startup scripts for services like DNS. This is helpful should your system hang when trying to start such services. Networking will be disabled as well as any multiuser access. Also you can use **linux -s** at the LILO prompt to enter runlevel 1. Should you want to enter the single-user state (runlevel 1) and also run the startup scripts, you can use the special **s** or **S** runlevels.

When your system starts up, it uses the default runlevel as specified in the default init entry in the **/etc/inittab** file. For example, if your default init runlevel is 5 (the graphical login), the default init entry in the **/etc/inittab** file would be

```
init:5:default
```

You can change the default runlevel by editing the **/etc/inittab** file and changing the **init** default entry. Editing the **/etc/inittab** file can be dangerous. You should do this with great care. As an example, if the default runlevel is 3 (command line), the entry for your default runlevel in the **/etc/inittab** file should look like the following:

```
id:3:initdefault:
```

You can change the 3 to a 5 to change your default runlevel from the command line interface (3) to the graphical login (5). Change only this number and nothing else.

```
id:5:initdefault:
```

Should your **/etc/inittab** file become corrupted, you can reboot and enter **linux single** at the boot prompt to start up your system, bypassing the **inittab** file. You can then edit the file to fix it.

No matter what runlevel you start in, you can change from one runlevel to another with the **telinit** command. If your default runlevel is 3, you power up in runlevel 3, but you can change to, say, runlevel 5 with **telinit 5**. In the next example, the **telinit** command changes to runlevel 1, the administrative state:

```
# telinit 1
```

telinit is really a link to the **init** command. It is the **init** command that performs that actual startup operation and is automatically invoked when your system starts up. Though you could use **init** to also change runlevels, it is best to use **telinit**. When invoked as **telinit**, **init** functions to merely change runlevels.

Tip You can use Linuxconf or Webmin to change the default startup runlevel between the graphical loginrunlevel (5) and a command line runlevel (text mode) (3).

Use the **runlevel** command to see what state you are currently running in. It will list the previous state followed by the current one. If you have not changed states, the previous state will be listed as N, indicating no previous state. This is the case for the state you boot up in. In the next example, the system is running in state 3, with no previous state change.

```
# runlevel
N 3
```

Although you can power down the system with the **telinit** command and the 0 state, you can also use the **shutdown** command. The **shutdown** command has a time argument that gives users on the system a warning before you power down. You can specify an exact time to shut down or a period of minutes from the current time. The exact time is specified by *hh***:***mm* for the hour and minutes. The period of time is indicated by a + and the number of minutes. The **shutdown** command takes several options with which you can specify how you want your system shut down. The **-h** option, which stands for halt, simply shuts down the system, whereas the **-r** option shuts down the system and then reboots it. In the next example, the system is shut down after ten minutes. The shutdown options are listed in Table 28-3.

```
# shutdown -h +10
```

Table 28-3: System Shutdown Options

Command	**Description**
shutdown [-rkhncft] *time* [*warning-message*]	Shuts the system down after the specified time period, issuing warnings to users; you can specify a warning message of your own after the time argument; if neither **-h** nor **-r** is specified to shut down the system, the system sets to the administrative mode, runlevel state 1.
Argument	
Time	Has two possible formats: it can be an absolute time in the format *hh*:*mm*, with *hh* as the hour (one or two digits) and *mm* as the minute (in two digits); it can also be in the format +*m*, with *m* as the number of minutes to wait; the word **now** is an alias for **+0**.
Option	
-t *sec*	Tells **init** to wait *sec* seconds for the interval between sending processes the warning and the kill signals, before changing to another runlevel.
-k	Doesn't actually shut down; only sends the warning messages to everybody.
-r	Reboots after shutdown, runlevel state 6.
-h	Halts after shutdown, runlevel state 0.
-n	Doesn't call **init** to do the shutdown; you do it yourself.
-f	Does a *fast* reboot.
-c	Cancels an already running shutdown; no time argument.

To shut down the system immediately, you can use **+0** or the word **now**. The following example has the same effect as the CTRL-ALT-DEL method of shutting down your system, as described in Chapter 3. It shuts down the system immediately, and then reboots.

```
# shutdown -r now
```

With the **shutdown** command, you can include a warning message to be sent to all users currently logged in, giving them time to finish what they are doing before you shut them down.

```
# shutdown -h +5 "System needs a rest"
```

If you do not specify either the **-h** or the **-r** options, the **shutdown** command shuts down the multiuser mode and shifts you to an administrative single-user mode. In effect, your system state changes from 3 (multiuser state) to 1 (administrative single-user state). Only the root user is active, allowing the root user to perform any necessary system administrative operations with which other users might interfere.

Note You can also shut down your system from the Gnome or KDE desktops.

System Directories and Files

Your Linux system is organized into directories whose files are used for different system functions. Directories with "bin" in the name are used to hold programs. The **/bin** directory holds basic user programs, such as login, shells (bash, tcsh, and zsh), and file commands (**cp**, **mv**, **rm**, **ln**, and so on). The **/sbin** directory holds specialized system programs for such tasks as file system management (fsck, fdisk, mkfs) and system operations like shutdown and startup (**lilo**, **init**). The **/usr/bin** directory holds program files designed for user tasks. The **/usr/sbin** directory holds user-related system operation, such as **useradd** to add new users. The **/lib** directory holds all the libraries your system makes use of, including the main Linux library, **libc**, and subdirectories such as **modules**, which holds all the current kernel modules.

```
# ls /
bin boot dev etc home lib lost+found mnt proc root sbin tmp usr var
```

The **/etc** directory holds your system, network, server, and application configuration files. Here you can find the **fstab** file listing your file systems, the **hosts** file with IP addresses for hosts on your system, and **lilo.conf** for the boot systems provided by LILO. This directory includes various subdirectories, such as **apache** for the Apache Web server configuration files and **X11** for the X Window System and window manager configuration files.

The **/mnt** directory is usually used for mount points for your CD-ROM, floppy, or Zip drives. These are file systems you may be changing frequently, unlike partitions on fixed disks. The **/home** directory holds user home directories. When a user account is set up, a home directory for it is set up here, usually with the same name as the user. On many systems, the **/home** directory also holds server data directories, such as **/ho**me/httpd for the Apache Web server Web site files or **/home/ftpd** for your FTP site files. The **/var** directory holds subdirectories for tasks whose files change frequently, such as lock files, log files, or printer spool files. Red Hat currently places its server data directories in the **/var** directory. The **/tmp** directory is simply a directory to hold any temporary files programs may need to perform a particular task.

The **/usr/src** directory holds source files; in particular, **/usr/src/linux** holds the kernel source files you use to update the kernel. Data that is meant to be shared by different applications is kept in the **/usr/share** directory. For example, the **/usr/share/doc** directory holds documentation that is usually installed with different applications. Here, you can also find HOW-TO documents. The **/usr /local** directory is used for programs meant to be used only on this particular system. The **/usr/opt** directory is where optional packages are installed.

```
# ls /usr
X11R6 bin cgi-bin dict doc etc games include info lib libexec local
man sbin share src tmp
```

Standard system directories and configuration files are shown in Tables 28-4 and 28-5. See Chapter 36 for network configuration files.

Table 28-4: System Directories

Directory	**Description**
/bin	System-related programs
/sbin	System programs for specialized tasks
/lib	System libraries
/etc	Configuration files for system and network services and applications
/home	The location of user home directories and server data directories, such as Web and FTP site files
/mnt	The location where CD-ROM and floppy disk files systems are mounted
/var	The location of system directories whose files continually change, such as logs, printer spool files, and lock files
/usr	User-related programs and files. Includes several key subdirectories, such as **/usr/bin**, **/usr/X11**, and **/usr/doc**
/usr/bin	Programs for users
/usr/X11	X Window System configuration files
/usr/share	Shared files
/usr/share/doc	Documentation for applications
/tmp	Directory for system temporary files

Table 28-5: Configuration Files

File	**Description**
/etc/inittab	Sets the default state, as well as terminal connections
/etc/passwd	Contains user password and login configurations
/etc/shadow	Contains user-encrypted passwords
/etc/group	Contains a list of groups with configurations for each
/etc/fstab	Automatically mounts file systems when you start your system
/etc/lilo.conf	The LILO configuration file for your system
/etc/conf.modules	Modules on your system to be automatically loaded
/etc/printcap	Contains a list of each printer and its specifications

Table 28-5: Configuration Files

File	Description
/etc/termcap	Contains a list of terminal type specifications for terminals that could be connected to the system
/etc/gettydefs	Contains configuration information on terminals connected to the system
/etc/skel	Directory that holds the versions of initialization files, such as **.bash_profile,** which are copied to new users' home directories
/etc/ttys	List of terminal types and the terminal devices to which they correspond
/etc/services	Services run on the system and the ports they use
/etc/profile	Default shell configuration file for users
/etc/shells	Shells installed on the system that users can use
/etc/motd	System administrator's message of the day

System Startup Files: /etc/rc.d and /etc/sysconfig

Each time you start your system, it reads a series of startup commands from system initialization files located in your **/etc/rc.d** directory. These initialization files are organized according to different tasks. Some are located in the **/etc/rc.d** directory itself, while others are located in a subdirectory called **init.d**. You should not have to change any of these files. The organization of system initialization files varies among Linux distributions. The Red Hat organization is described here. Some of the files you find in **/etc/rc.d** are listed in Table 28-6.

Table 28-6: System Startup Files

File	Description
/etc/sysconfig	Directory on Red Hat Linux that holds system configuration files and directories.
/etc/rc.d	Directory that holds system startup and shutdown files.
/etc/rc.d/rc.sysinit	Initialization file for your system.
/etc/rc.d/rc.local	Initialization file for your own commands; you can freely edit this file to add your own startup commands; this is the last startup file executed.
/etc/rc.d/rc.modules	Loads kernel modules (not implemented by default on Red Hat Linux).
/etc/rc.d/init.d	Directory that holds many of the daemons, servers, and scripts such as **httpd** for Web servers and networks to start up network connections.
/etc/rc.d/rc*num***.d**	Directories for different runlevels where *num* is the runlevel. The directories hold links to scripts in the **/etc/rc.d/init.d** directory.
/etc/rc.d/init.d/halt	Operations performed each time you shut down the system, such as unmounting file systems; called **rc.halt** in other distributions.

Table 28-6: System Startup Files

File	Description
/etc/rc.d/init.d/lpd	Start up and shut down the **lpd** daemon.
/etc/rc.d/init.d/inet	Operations to start up or shut down the **inetd** daemon.
/etc/rc.d/init.d/network	Operations to start up or shut down your network connections.
/etc/rc.d/init.d/httpd	Operations to start up or shut down your Web server daemon, **httpd**.

The **/etc/rc.d/rc.sysinit** file holds the commands for initializing your system, including the mounting of your file systems. Kernel modules for specialized features or devices can be loaded in an **rc.modules** file. The **/etc/rc.d/rc.local** file is the last initialization file executed. You can place commands of your own here. If you look at this file, you see the message displayed for you every time you start the system. You can change that message if you want. When you shut down your system, the **halt** file, which contains the commands to do this, is called. The files in **init.d** are then called to shut down daemons, and the file systems are unmounted. In the current distribution of Red Hat, **halt** is located in the **init.d** directory. For other distributions, it may be called **rc.halt** and located in the **/etc/rc.d** directory.

The **/etc/rc.d/init.d** directory is designed primarily to hold scripts that both start up and shut down different specialized daemons. Network and printer daemons are started up here. You also find files here to start font servers and Web site daemons. These files perform double duty, starting a daemon when the system starts up and shutting down the daemon when the system shuts down. The files in **init.d** are designed in a way to make it easy to write scripts for starting up and shutting down specialized applications. It uses functions defined in the **functions** file, as do many of the other **init.d** files. Many of these files are set up for you automatically. You needn't change them. If you do change them, be sure you know how these files work first. Chapter 22 describes this process in detail.

When your system starts up, several programs are automatically started and run continuously to provide services such as Web site operations. Depending on what kind of services you want your system to provide, you can add or remove items in a list of services to be automatically started. In the installation process, you could determine what services those would be. For example, the Web server is run automatically when your system starts up. If you are not running a Web site, you would have no need, as yet, for the Web server. You could have the service not started, removing an extra task the system does not need to perform. Several of the servers and daemons perform necessary tasks. The **sendmail** server enables you to send messages across networks, while the **lpd** server performs printing operations.

When your system starts up, it uses links in special runlevel directories in the **/etc/rc.d/** directory to run the startup scripts in the **/etc/rc.d/init.d** directory. A runlevel directory bears the number of its runlevel, as in **/etc/rc.d/rc3.d** for runlevel 3, and **/e**tc/rc.d/rc5.d for runlevel 5. To have a service not start up, remove its link from that runlevel directory. You can use any of these scripts to start and stop a daemon manually at any time by using the **stop** argument to stop it, the **start** argument to start it again, and the **restart** argument to restart the daemon.

Most administration tools provide interfaces displaying a simple list of services from which you can select the ones you want to start up. On the Red Hat Setup menu, select System Services and then choose from the list of servers and daemons provided. Toggle an entry on or off with the SPACEBAR. On Linuxconf, the Control Service Activity panel lists different daemons and servers that you can have start by just clicking a check box. On Webmin, select Bootup and Shutdown on the System panel to display a list of available services that you can then configure to start up.

In addition, you can use a KDE System V Init Editor (**ksysv**) (see Chapter 22) to determine which servers and daemons are to start and stop at what runlevel. **ksysv** provides an easy-to-use GUI interface for managing the servers and daemons in your **/etc/rc.d/init.d** directory. You can stop, start, and assign servers to different runlevels. **ksysv** is easier to use because it supports drag-and-drop operations. To assign a server to a particular runlevel, drag its entry from the Services box to the appropriate Runlevel box. To remove it from a particular runlevel, drag its entry out of that Runlevel box to the Trash icon. To start and stop a daemon manually, right-click it and select either the Stop or Start entry from the pop-up menu.

Note From the Red Hat Control Panel, you can also run the System V Runlevel Editor, which operates similarly to the KDE System V Init Editor.

On Red Hat systems, configuration and startup information is also kept in the **/etc/sysconfig** directory. Here you will find files containing definitions of system variables used to configure devices such as your keyboard and mouse. These entries were defined for you when you configured your devices during installation. You will also find network definitions as well as scripts for starting and stopping your network connections. A sample of the keyboard file is shown here.

/etc/sysconfig/keyboard

```
KEYBOARDTYPE="pc"
KEYTABLE="us"
```

Several of these files are generated by Red Hat configuration tools such as **mouseconfig** or **netconfig** (see Chapter 29 for the specific files these tools control). For example, **mouseconfig** will generate configuration variables for the mouse device name, type, and certain features, placing them in the **/etc/sysconfig/mouse** file, shown here:

```
FULLNAME="Generic - 3 Button Mouse (USB)"
MOUSETYPE="imps2"
XEMU3="no"
XMOUSETYPE="IMPS/2"
```

Other files like **hwconf** will list all your hardware devices, defining configuration variables such as its class (video, CD-ROM, hard drive, etc.) the bus it uses (PCI, IDE, etc.), its device name (such as hdd or st0), the drivers it uses, and a description of the device. A CD-ROM entry is shown here:

```
class: CDROM
bus: IDE
detached: 0
device: hdd
driver: ignore
desc: "TOSHIBA DVD-ROM SD-M1402"
```

Several directories are included, such as **network-scripts**, which list several startup scripts for network connections-such as **ifup-ppp**, which starts up PPP connections.

System Logs: /var/log and syslogd

Various system logs kept for tasks performed on your system are kept in the **/var/log** directory. Here you can find logs for mail, news, and all other system operations. The **/var/log/messages** file is a log of all system tasks not covered by other logs. This usually includes startup tasks, such as loading drivers and mounting file systems. If a driver for a card failed to install at startup, you find an error message for it here. Logins are also logged in this file, showing you who attempted to log into what account. The **/var/log/maillog** file logs mail message transmissions and news transfers.

Logs are managed by the **syslogd** daemon. This daemon will manage all the logs on your system as well as coordinating with any the logging operations of other systems on your network. Configuration information for **syslogd** is held in the **/etc/syslog.conf** file. This file contains the names and locations for your system log files. Here you find entries for **/var/log/messages** and **/var/log/maillog**, among others. An entry consists of two fields: a selector and an action. The selector is the kind of service to be logged, such as mail or news, and the action is the location where messages are to be placed. The action is usually a log file, but it can also be a remote host or a pipe to another program. The kind of service is referred to as a facility. **syslogd** has several terms it uses to specify certain kinds of service (see Table 28-7). A facility can be further qualified by a priority. A priority specifies the kind of message generated by the facility. **syslogd** uses several designated terms to indicate different priorities. A sector is constructed from both the facility and priority, separated by a period. For example, to save error messages generated by mail systems, you use a sector consisting of the mail facility and the **err** priority, as shown here:

```
mail.err
```

Table 28-7: Syslogd Facilities, Priorities, and Operators

Facility	**Description**
auth-priv	Security/authorization messages (private)
cron	Clock daemon (**cron** and **at**) messages
daemon	Other system daemon messages
kern	Kernel messages
lpr	Line printer subsystem messages
mail	Mail subsystem messages
mark	Internal use only
news	Usenet news subsystem messages
syslog	Syslog internal messages
user	Generic user-level messages
uucp	UUCP subsystem messages
local0 through local7	Reserved for local use
Priority	
debug	7, Debugging messages, lowest priority

Table 28-7: Syslogd Facilities, Priorities, and Operators

Facility	**Description**
info	6, Informational messages
notice	5, Notifications, normal, but significant, condition
warning	4, Warnings
err	3, Error messages
crit	2, Critical conditions
alert	1, Alerts, action must be taken immediately
emerg	0, Emergency messages, system is unusable, highest priority
Operator	
*	Matches all facilities or priorities in a sector
=	Restrict to a specified priority
!	Exclude specified priority and higher ones
Operator	
/	A file to save messages to
@@	A host to send messages to
|	FIFO pipe to send messages to

To save these messages to the **/var/log/maillog** file, you specify that file as the action, giving you the following entry:

```
mail.err /var/log/maillog
```

syslogd also supports the use of ***** as a matching character to match either all the facilities or priorities in a sector. **cron.*** would match on all cron messages no matter what the priority, ***.err** would match on error messages from all the facilities, and ***.*** would match on all messages. The following example saves all mail messages to the **/var/log/maillog** file and all critical messages to the **/var/log/mycritical** file:

```
mail.* /var/log/maillog
*.crit /var/log/mycritical
```

When you specify a priority for a facility, that will in fact include all the messages with a higher priority. So the **err** priority also includes the **crit**, **alret**, and **emerg** priorities. If you just want to select the message for a specific priority, you qualify the priority with the = operator. For example, **mail.=err** will select only error messages, not **crit**, **alert**, and **emerg** messages. You can also restrict priorities with the **!** operator. This will eliminate messages with the specified priority and higher. For example, **m**ail.!crit will exclude **crit** messages and the higher **alert** and **emerg** messages. To specifically exclude all the messages for an entire facility, you use the **none** priority. **mail.none** excludes all mail messages. This is usually used when you are defining several sectors in the same entry.

You can list several priorities or facilities in a given sector by separating them with commas. You can also have several sectors in the same entry by separating them with semicolons. The first example saves to the **/var/log/messages** file all messages with **i**nfo priority, excluding all mail, news, and authentication messages (**authpriv**). The second saves all **crit** messages and higher for the **uucp** and **news** facilities to the **/var/log/spooler** file:

```
*.info;mail.none;news.none;authpriv.none /var/log/messages
uucp,news.crit /var/log/spooler
```

For the action field, you can specify files, remote systems, users, or pipes. An action entry for a file must always begin with a / and specify its full path name, such as **/var**/log/messages. To log messages to a remote host, you simply specify the hostname preceded by an **@** sign. The following example saves all kernel messages on **rabbit.trek.com**:

```
kern.* @rabbit.trek.com
```

For users, you just list the login names of the users you want to receive the messages. The following example will send critical news messages to the consoles for the users **chris** and **aleina**:

```
news.=crit chris,aleina
```

You can also output messages to a named pipe (FIFO). The pipe entry for the action field begins with a **|**. The following example pipes kernel debug messages to the name pipe **|/usr/adm/debug**:

```
kern.=debug |/usr/adm/debug
```

Whenever you make changes to the **syslog.conf** file, you need to restart the **syslogd** daemon using the following command:

```
/etc/rc.d/init.d/syslog restart
```

The default **/etc/syslog.conf** file for Red Hat systems is shown here. Messages are logged to various files in the **/var/log** directory.

/etc/syslog.conf

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                         /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;news.none;authpriv.none   /var/log/messages

# The authpriv file has restricted access.
authpriv.* /var/log/secure

# Log all the mail messages in one place.
mail.* /var/log/maillog

# Everybody gets emergency messages, plus log them on another
# machine.
*.emerg *

# Save mail and news errors of level err and higher in a
# special file.
```

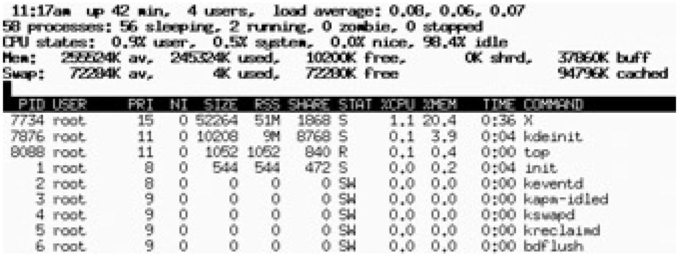
```
uucp,news.crit /var/log/spooler

# Save boot messages also to boot.log
local7.* /var/log/boot.log

#
# INN
#
news.=crit /var/log/news/news.crit
news.=err /var/log/news/news.err
news.notice /var/log/news/news.notice
```

Performance Analysis Tools and Processes

Each task performed on your system is treated by Linux as a process, and is assigned a number and a name. You can examine these processes and even stop them. From the command line, you can use the **ps** command to list processes. With the **-aux** command, you can list all processes. Piping the output to a **grep** command with a pattern enables you to search for a particular process. The following command lists all X Window System processes:

```
ps -aux | grep 'X'
```

A number of utilities on your system provide detailed information on your processes, as well as other system information such as CPU and disk use (see Table 28-8). Although these tools were designed to be used on a shell command line, displaying output in text lines, several now have KDE and Gnome versions that provide a GUI interface for displaying results and managing processes. The **vmstat** command outputs a detailed listing indicating the performance of different system components, including CPU, memory, I/O, and swap operations. A report is issued as a line with fields for the different components. If you provide a time period as an argument, it repeats at the specified interval-usually a few seconds. The **top** command provides a listing of the processes on your system that are the most CPU intensive, showing what processes are using most of your resources. The listing is in real time and updated every few seconds. Commands are provided for changing a process's status, such as its priority (see Figure 28-2).



```
 11:17am  up 42 min,  4 users,  load average: 0.08, 0.06, 0.07
58 processes: 56 sleeping, 2 running, 0 zombie, 0 stopped
CPU states:  0.9% user,  0.5% system,  0.0% nice, 98.4% idle
Mem:   255524K av,  245324K used,   10200K free,       0K shrd,   37860K buff
Swap:   72284K av,       4K used,   72280K free                   94796K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
 7734 root      15   0 52264  51M  1868 S     1.1 20.4   0:36 X
 7876 root      11   0 10208   9M  8768 S     0.1  3.9   0:04 kdeinit
 8088 root      11   0  1052 1052   840 R     0.1  0.4   0:00 top
    1 root       8   0   544  544   472 S     0.0  0.2   0:04 init
    2 root       8   0     0    0     0 SW    0.0  0.0   0:00 keventd
    3 root       9   0     0    0     0 SW    0.0  0.0   0:00 kapm-idled
    4 root       9   0     0    0     0 SW    0.0  0.0   0:00 kswapd
    5 root       9   0     0    0     0 SW    0.0  0.0   0:00 kreclaimd
    6 root       9   0     0    0     0 SW    0.0  0.0   0:00 bdflush
```

Figure 28-2: The **top** command

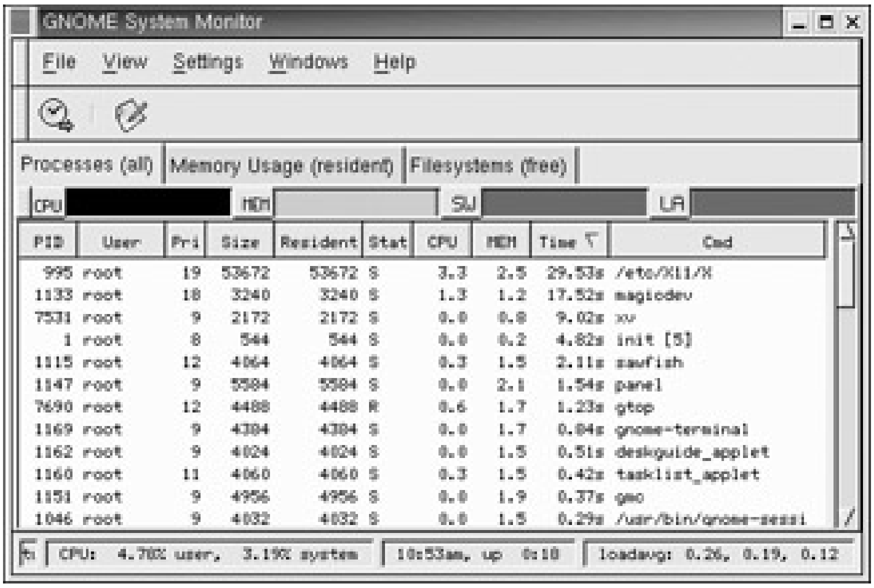
The **free** command lists the amount of free RAM memory on your system, showing how much is used and how much is free, as well as what is used for buffers and swap memory. **Xload** is an X Window System tool showing the load, CPU, and memory. **iostat** will display your disk usage, and **sar** will show system activity information (see Table 28-8).

Table 28-8: Performance Tools

Performance Tool	Description

Table 28-8: Performance Tools	
Performance Tool	**Description**
vmstat	Performance of system components
top	Listing of most CPU intensive processes
free	Listing for free RAM memory
sar	System activity information
iostat	Disk usage
Xload	View of system load
Gtop	Gnome System Manager
Kpm	KDE Process Manager

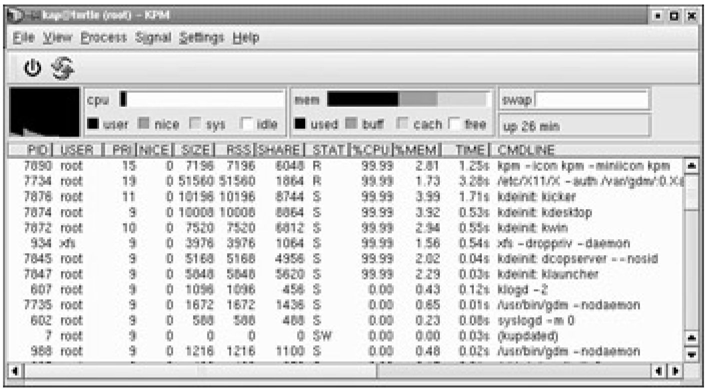
On the Gnome System Manager (GTop), you can also sort the processes according to their fields by clicking the field's button at the top of the process list. If you right-click an entry, a pop-up menu displays with actions you can perform on it (see Figure 28-3). System statistic summary graphs are displayed at the top of the window showing the CPU load, memory use, and disk use. You can add more graphs or change their display features, such as the colors used. The GTop window displays three tabbed panels for detailed reports showing processes, memory use, and file system use. You can add more, showing customized reports such as only the user processes. Process lists can be further refined to show user, system, or all processes. To configure Gtop, you select the Preferences entry in the Settings menu. This displays a menu with tabbed panels for specifying the update frequency for different statistics, determining the summaries you want displayed and what process fields to show. You can find the Gnome System Manager in the Utilities menu.



```
  PID User      Pri  Size  Resident Stat  CPU   MEM   Time   Cmd
  995 root       19  53672    53672 S     3.3   2.5  29.53s /etc/X11/X
 1133 root       18   3240     3240 S     1.3   1.2  17.52s magiodev
 7531 root        9   2172     2172 S     0.0   0.8   9.02s xv
    1 root        8    544      544 S     0.0   0.2   4.82s init [5]
 1115 root       12   4064     4064 S     0.3   1.5   2.11s sawfish
 1147 root        9   5584     5584 S     0.0   2.1   1.54s panel
 7690 root       12   4488     4488 R     0.6   1.7   1.23s gtop
 1169 root        9   4384     4384 S     0.0   1.7   0.84s gnome-terminal
 1162 root        9   4024     4024 S     0.0   1.5   0.51s deskguide_applet
 1160 root       11   4060     4060 S     0.3   1.5   0.42s tasklist_applet
 1151 root        9   4956     4956 S     0.0   1.9   0.37s gmc
 1046 root        9   4032     4032 S     0.0   1.5   0.29s /usr/bin/gnome-sessi

CPU:  4.70% user,  3.19% system | 10:53am, up  0:10 | loadavg: 0.26, 0.19, 0.12
```

Figure 28-3: Gnome system monitor

The K Desktop provides the KDE Process Manager (kpm) for viewing and managing your processes (see in Figure 28-4). You can sort the processes according to their fields by clicking the field's button at the top of the process list. If you select a process, you can then choose to perform several different actions on it, such as ending it (killing the process) or suspending it (putting it to sleep). A right-click on a process entry displays a pop-up menu with the different actions you can take. You can further refine your process list by choosing to view only your own processes, system processes, or all processes.


kap@turtle (root) - KPM
File View Process Signal Settings Help
cpu
■ user ▨ nice □ sys □ idle
mem
■ used ▨ buff □ cach □ free
swap
up 26 min

```
  PID USER   PRI NICE  SIZE   RSS SHARE STAT  %CPU %MEM  TIME  CMDLINE
 7890 root    15    0  7196  7196  6048 R    99.99 2.81 1.25s kpm -icon kpm -miniicon kpm
 7734 root    19    0 51560 51560  1864 R    99.99 1.73 3.28s /etc/X11/X -auth /var/gdm/:0.X
 7876 root    11    0 10196 10196  8744 S    99.99 3.99 1.71s kdeinit: kicker
 7874 root     9    0 10008 10008  8864 S    99.99 3.92 0.53s kdeinit: kdesktop
 7872 root    10    0  7520  7520  6812 S    99.99 2.94 0.55s kdeinit: kwin
  934 xfs      9    0  3976  3976  1064 S    99.99 1.56 0.54s xfs -droppriv -daemon
 7845 root     9    0  5168  5168  4956 S    99.99 2.02 0.04s kdeinit: dcopserver --nosid
 7847 root     9    0  5848  5848  5620 S    99.99 2.29 0.03s kdeinit: klauncher
  607 root     9    0  1096  1096   456 S     0.00 0.43 0.12s klogd -2
 7735 root     9    0  1672  1672  1436 S     0.00 0.65 0.01s /usr/bin/gdm -nodaemon
  602 root     9    0   588   588   488 S     0.00 0.23 0.08s syslogd -m 0
    7 root     9    0     0     0     0 SW    0.00 0.00 0.03s (kupdated)
  988 root     9    0  1216  1216  1100 S     0.00 0.48 0.02s /usr/bin/gdm -nodaemon
```

Figure 28-4: The KDE Process Manager

Chapter 29: Configuration Tools and Boot Management

Overview

Although system administration can become complex, basic administration tasks such as configuring devices and setting up your network are easy to perform, particularly if you use a configuration tool such as the Text Mode Setup utility. Red Hat provides several specialized older tools such as netcfg for network configuration and TimeTool for setting the system date and time. Most of the older Red Hat administration tools can be accessed through entries on the Gnome desktop system menu. The KDE desktop also includes administrative tools such as Kuser to manage users and the Boot Manager to select operating systems to start up. These tools have been discussed as their topics have been covered in various chapters. This chapter will cover the Red Hat Text Mode Setup Utility and the Linux boot management utilities known as the Linux Loader (LILO) and the Grand Unified Bootloader (GRUB).

Tip In addition to those tools already provided with Red Hat, you can also download and install comprehensive administrative tools like Linuxconf or Webmin. These tools will perform almost all administrative tasks, eliminating the need to go hunting around for tools to perform different tasks.

All administrative tools are operated as front ends for making entries in Linux configuration files. You can edit these files and make entries directly, if you want. The underlying administration tasks are the same.

With Linux, you also have the ability to load up different versions of the Linux kernel as well as different operating systems that you have installed on your system. The task of selecting and starting up an operating system or kernel is managed by a boot management utility called the Linux Loader, also known simply as LILO. LILOis a versatile tool, letting you load operating systems that share the same disk drive, as well as letting you choose from different Linux kernels that may be installed on the same Linux system.

Red Hat Text Mode Setup Utility

Red Hat provides the Text Mode Setup utility (Setuptool) with which you can configure different devices and system settings, such as your keyboard, mouse, and time zone. It has a cursor-based interface that you run from the command line, using arrow keys, the SPACEBAR, and the ENTER key to make choices. Setuptool is useful if you have changed any of your devices-say, installed a new mouse, keyboard, or sound card. Start the Setuptool utility with the command **setup**, which you enter at a shell command line. On the desktop, you can open a terminal window and enter the command, or select the Text Mode Setup menu entry in the Gnome system menu. A menu of configuration choices is displayed. The utility will run within a terminal window, as if you were running it from the command line. Use the arrow keys to select one, and then press the TAB key to move to the Run Tool and Quit buttons. Figure 29-1 shows the initial Setup menu.

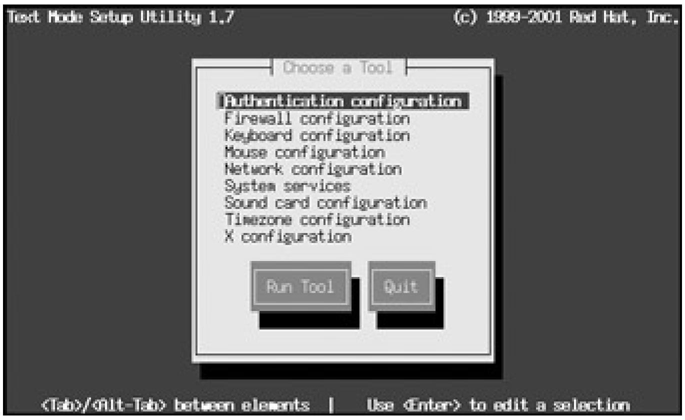


Figure 29-1: Red Hat Text Mode Setup utility

The Setuptool utility is actually an interface for running several configuration tools (see Table 29-1). You can call any of these tools separately using their commands. For example, the **kbdconfig** command starts the keyboard configuration utility that enables you to select the type of keyboard, while the **mouseconfig** command enables you to select the type of mouse. These tools will generate configuration files defining variables for use in enabling devices and services. Most are found in the **/etc/sysconfig** directory. For example, **mouseconfig** defines configuration variables that it then saves to the **/etc/sysconfig/mouse** file.

Table 29-1: Setup Tools

Tool	**Configuration File**	**Description**
setup		Red Hat Setuptool interface listing configuration tools for system and device setting.
authconfig	**/etc/sysconfig/authconfig** **/etd/sysconfig/network**	Authentication options, such as enabling NIS, shadow passwords, and MD5 passwords.
lokkit	**/etc/sysconfig/ipchains**	Selects the level of firewall protection: High, Medium, and None. **gnome-lokkit** runs a Gnome interface.
kbdconfig	**/etc/sysconfig/keyboard**	Selects the keyboard type.
mouseconfig	**/etc/sysconfig/mouse**	Selects the mouse type.

Table 29-1: Setup Tools

Tool	**Configuration File**	**Description**
netconfig	**/etc/sysconfig/network** **/etc/sysconfig/network-scripts/ifcfg-eth*N***	Sets your LAN network settings.
ntsysv	**/etc/sysconfig/auth**	Selects servers and daemons to start up at boot time.
sndconfig	**/etc/sysconfig/soundcard**	Detects and configures your sound card.
timeconfig	**/etc/sysconfig/clock** **/etc/sysconfig/localtime**	Selects the time zone.
Xconfigurator	**/etc/X11/XF86Config**	Configures your X Window System for your video card and monitor.

authconfig is used to enable authentication services such as NIS, Kerberos, and LDAP. It displays a dialog box for enabling the Network Information Service (NIS), password encryption, and LDAP support. You find an entry for NIS, and entries for the shadow and MD5 password security methods. **authconfig** places its configuration information in several files. The **etc/sysconfig/authconfig** file specifies whether certain authentication services are enabled. It will further configure configuration files for the services selected, such as **/etc/krb5.conf** for Kerberos authentication and **/etc/yp.conf** for NIS support, as well as **/etc/openldap/ldap.conf** for LDAP authentication.

With **lokkit**, you can select the level of firewall protection at High, Medium, or None. An advanced option lets you select different services to be allowed such as mail, the secure shell, FTP, the Web server, and the FTP server. You can also run **gnome-lokkit**, which will prompt you for different settings using Gnome dialog boxes. **Lokkit** currently provides firewall rules only for the older IP-Chains firewall service. It does not support the newer IP-Tables. **lokkit** places its IP-Chains firewalling rules in the **/etc/sysconfig/ipchains** file.

With **kbdconfig**, you can select the type of keyboard you are using. A cursor-based dialog box appears with a list of different keyboard types. **kbdconfig** places its configuration information in the **etc/sysconfig/keyboard** file.

With **mouseconfig**, you can select the type of mouse you are using. A cursor-based dialog box appears with a list of different mouse device types. Your system is automatically probed for the type of mouse connected to your system, and the cursor is positioned at that entry. If you have a two-button mouse, you can select the three-button emulation option to let a simultaneous click on both the left and the right mouse buttons emulate a third mouse button. **kbdconfig** places its configuration information in the **etc/sysconfig/mouse** file. It will also set the **/dev/mouse** symbolic link to the mouse device file.

With **netconfig**, you can enter the LAN setting for your network connections. This tool will not help for modem connections. Here you can enter basic LAN settings like your IP address, gateway address, netmask, and DNS server addresses. **netconfig** places its network configuration information like the hostname and gateway in the **etc/sysconfig/network** file. Specific Ethernet device configurations, which would include your IP address and netmask, are place in the appropriate Ethernet device configuration file in the **/etc/sysconfig/network-scripts** directory. For example, the IP address and netmask used for the eth0 Ethernet device can be found in **/etc/sysconfig/network-scripts/ifcfg-eth0**. Local host settings are in **/etc/sysconfig/network-scripts/ifcfg-lo.**

The ntsysv utility is a simple utility for specifying which servers and services should be automatically started at boot time (see Chapter 15). The dialog box lists the possible servers and services to choose from. Move to the entry you want and use the SPACEBAR to toggle it on or off. An entry with an asterisk next to it is selected and is started automatically the next time you boot your system. ntsysv uses **chkconfig** to make its changes to the startup services.

The sndconfig utility enables you to select and configure your sound card. It initially tries to detect your sound card automatically. If the automatic detection fails, a dialog box appears with a listing of different sound cards. Select the one on your system. Another dialog box appears where you need to enter the setting for your sound card. sndconfig then tries to play sample sound and MIDI files to test the card. As an alternative to sndconfig, you can obtain, load, and configure sound drivers yourself. sndconfig places its configuration information in the **etc/sysconfig/soundcard** file. It will also store the appropriate module specification in the **/etc/modules.conf** file, as well as the **/etc/isapnp.conf** file if needed.

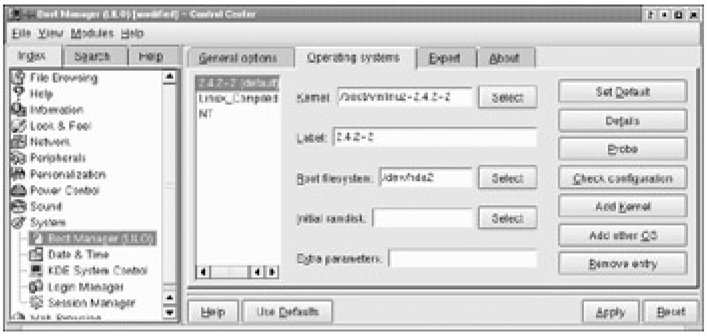
You can use the simple timeconfig utility to specify your system's time zone. This is useful if your system has moved to a different time zone. timeconfig places its configuration information in the **etc/sysconfig/clock** and **/etc/localtime** files.

One important utility is the X Window System setup provided by Xconfigurator. If you have trouble with your X Window System configuration and can no longer start up your graphical interface, you can use the **Xconfigurator** command to configure it from the command line. You could switch to or start up in runlevel 3, the text mode, and then run Xconfigurator entirely from the command line, without ever having to run the X Window System. Xconfigurator is also helpful for updating X Windows if you change your video card. Run Xconfigurator again and select the card. When you download a new version of XFree86, the packages includes a compatible version of Xconfigurator. Xconfigurator will generate the X Window System **/etc/X11/XF86Config** configuration file and use the specifications in **/etc/sysconfig/mouse** to configure your X Window System mouse.

LILO

If you have installed two or more operating systems on your computer's hard disks, you need to use a boot manager to enable you to choose the one you want to use whenever your computer starts up. Most Linux distributions provide the Linux Loader (LILO) as its boot manager, which you can install as part of the installation process.

You can modify your LILO configuration either by using an administration tool like Boot Manger (LILO-config), or by editing the **/etc/lilo.conf** configuration file directly. If it is installed, you can access Boot Manager on the KDE desktop Control Center, under the System entry. The Boot Manager will display the four panels: General Options, Operating Systems, Expert, and About. The General panel lets you set basic options such as the drive for the boot record. On the Operating System panel you create the entries for different operating systems on your computer (see Figure 29-2). For Linux systems, you can specify the root partition and the Linux kernel, as well as a label. The Expert panel lets you edit the **lilo.conf** file directly, letting you type in options and add stanzas.



2.4.2-2 (default)
Linux_Compiled
NT

Kernel: /boot/vmlinuz-2.4.2-2

Label: 2.4.2-2

Boot filesystem: /dev/hda2

Initial ramdisk:

Extra parameters:

Figure 29-2: Boot Manager (KDE Control Center, System)

Note You can also configure LILO with Webmin and Linuxconf, or with the KDE Klilo2 tool.

You can directly modify your LILO configuration by editing the **/etc/lilo.conf** configuration file and executing the command **lilo.** If you examine your **/etc/lilo.conf** file, you find it organized into different segments called *stanzas*, one for each operating system that LILO is to start up. If your Linux system shares your computer with a Windows system, you should see two stanzas listed in your **/etc/lilo.conf** file: one for Linux and one for Windows. Each stanza indicates the hard disk partition on which the respective operating system is located. It also includes an entry for the label. This is the name you enter at the LILO prompt to start that operating system.

Entries in the **lilo.conf** file consist of options to which you assign values with the = operator (see Table 29-2). At the beginning of the file, you enter global options. The entries you need will already be generated for you if you have already set up LILO during installation. An example of a Linux global entry follows. To set the timeout period when LILO waits for you to make an entry before starting the default, you create a timeout entry and assign to it the number of seconds you want to wait.

```
timeout = 200
```

Since LILO is invoking your Linux system, you can specify any options you would normally pass to that system in the **lilo.conf** file. You do this with the append option. In effect, such options are appended to those you may enter manually at the boot prompt. Instead of manually entering the same options at the boot prompt, you can specify them in the **lilo.conf** file with the append option. You have already seen how this is done for IDE CD-R and CD-RW devices in Chapter 4. The following example configures the IDE CD-RW drive on the Secondary IDE master connection (hdc) as a SCSI device by loading the ide-scsi module for it.

```
append="hdc=ide-scsi"
```

Another useful global entry is **default**, with which you can set the default operating system to start. It takes as its argument the label for the stanza. In the following example, the stanza labeled **win** will be executed by default, in this case, starting Windows:

```
default = win
```

Then you start a Linux stanza with an **image** or **other** option. You use the **image** option for Linux boot images files, and the **other** option for other operating systems like Windows. The next example starts a stanza for Linux loading a Linux 2.4.2-2:

```
image = /boot/vmlinuz-2.4.2-2
```

You can then enter options for a stanza, such as its label and the partition that holds its root device:

```
label = linux
root = /dev/hda4
```

The **other** option will start a stanza for a non-linux system. It is used in the following example to reference a Windows operating system located on the first hard drive partition.

```
 other = /dev/hda1
```

One helpful option is **password**, where you can restrict access to a particular boot image file or operating system with a specified password. When the user selects that image or operating system, they are prompted for the password.

```
password = mypassword
```

Certain options are designed to be used only in the stanzas, and of those, some are to be used only for Linux image files or for other operating systems. For example, **table**, **label**, **password**, and **restricted** are used only for a particular stanza. They are not global options. The **append**, **literal**, **read-only**, and **read-write** options are to be used only in stanzas for Linux kernel images. The **append** and **literal** options are used to pass kernel image arguments to the kernel.

A stanza continues until another image or other option is reached, starting a new stanza. You can, if you want, make changes directly to the **/etc/lilo.conf** file using a text editor. Whenever you make a change, you must execute the **lilo** command to have it take effect. Type **lilo** and press ENTER.

```
# lilo
```

This command will read the **/etc/lilo.conf** and generate a corresponding boot record it will then write to your system's boot sector.

In the following **lilo.conf** example, the boot device is the hard drive labeled **/dev/hda**. This is the first IDE hard drive on a PC system. The timeout period where LILO waits for a user to enter a system's label is 200 seconds. The message it displays before the prompt is located in the file **/boot/message**. The user will be prompted (**prompt**) to enter a label. The default operating system (**default**) that will be started if the user does not enter one is the one labeled **win**-in this case Windows. The VGA mode (**vga**) that LILO will use is normal. The map (**map**) is located in the **/boot/map** file, and the **/boot/boot.p** is the file used as the new boot sector (**install**). The first stanza is a Linux image file (**image**) named **/boot/vmlinuz-2.4.2-2**. It is labeled "linux", and the root partition (**root**) for this Linux system is at **/dev/hda4**, the fourth partition on the first IDE hard drive. The Linux image will first be mounted as read only, and then later mounted as read/write. The second stanza denotes that there is another operating system (**other**) on the first partition in the first IDE hard drive. It is labeled "win". The partition table (**table**) for this operating system is on the first IDE hard drive.

/etc/lilo.conf

```
# general section
boot = /dev/hda
# wait 20 seconds (200 10ths) for user to select the entry to load
timeout = 200
message = /boot/message
prompt
 # default entry
default = win
vga = normal
 map=/boot/map
install=/boot/boot.b

image = /boot/vmlinuz-2.4.2-2
label = linux
 root = /dev/hda4
 read-only

other = /dev/hda1
label = win
 table=/dev/hda
```

Unless specified by the default entry, the default operating system LILO boots is the one whose segment is the first listed in the **lilo.conf** file. Because the Linux stanza is the first listed, this is the one LILO boots if you don't enter anything at the LILO prompt. If you want to have your Windows system be the default, you can use **lilo** with the **-D** option to reset the default, or you can edit the **lilo.conf** file to assign a value to **default**. You could also use a text editor to place the Windows stanza first, before the Linux stanza. Be sure to execute **lilo** to have the change take effect. The next time you start your system, you could press ENTER at the LILO prompt to have Windows loaded, instead of typing **win**.

```
# lilo -D win
```

You can set a number of LILO options using either command line options or making entries in the **lilo.conf** file. These options are listed in Table 29-2.

Table 29-2: LILO Options for the Command Line and lilo.conf Entries

Command Line Option	**lilo.conf Option**	**Description**
-u		Uninstall LILO, by copying the saved boot sector back.
-V		Print version number.
-t		Test only. Do not actually write a new boot sector or map file. Use together with **-v** to learn what LILO is about to do.
-I *label*		Display label and pathname of running kernel. Label is held in BOOT_IMAGE shell variable.
Global Option		

Table 29-2: LILO Options for the Command Line and lilo.conf Entries

Command Line Option	**lilo.conf Option**	**Description**
-b *bootdev*	**boot**=*bootdev*	Boot device.
-c	**compact**	Enable map compaction. Speed up booting.
-d *dsec*	**delay**=*dsec*	Timeout delay to wait for you to enter the label of an operating system at the LILO prompt when you boot up.
-D *label*	**default**=*label*	Use the kernel with the specified label, instead of the first one in the list, as the default kernel to boot.
-i *bootsector*	**install**=*bootsector*	File to be used as the new boot sector.
-f *file*	**disktab**=*file*	Disk geometry parameter file.
-l	**linear**	Generate linear sector addresses, instead of sector/head/cylinder addresses (for large hard disks. This option can cause a conflict with compact.)
-m *mapfile*	**map**=*mapfile*	Use specified map file instead of the default.
	message=*message-file*	Specify the message to be displayed before the LILO prompt.
	prompt	Displays the **lilo:** prompt. If timeout is not specified, the automatic reboots are disabled.
	timeout=*dsec*	Timeout delay to wait for you to enter the label of an operating system at the LILO prompt when you boot up.
-p fix	**fix-table**	Fix corrupt partition tables.
-P ignore	**ignore-table**	Ignore corrupt partition tables.
-s *file*	**backup**=*file*	Alternate save file for the boot sector.
-S *file*	**force-backup**=*file*	Allow overwriting of existing save file.
	vga=*mode*	VGA text mode that LILO uses.
Stanza Option		
	image=*Linux-kernel*	Pathname for boot image of a Linux kernel.
	other=*os-boot-image*	Pathname for boot image of a non-Linux operating system.
	alias=*name*	Another name for the kernel image or operating system label.
	table=*device*	Drive that holds the partition table.
	password=*password*	Protect access to an OS or kernel image with a password, as specified.

Table 29-2: LILO Options for the Command Line and lilo.conf Entries

Command Line Option	**lilo.conf Option**	**Description**
	restricted	User must enter a password if any command line parameters are passed to the kernel image.
-v	**verbose=***level*	Increase verbosity.
Linux Image Stanza Option		
	root=*path*	Partition for the kernel root.
	append=*string*	Arguments to append to arguments specified for the invocation of the kernel image. Used to add needed hardware specifications for a kernel image.
	literal=*string*	Arguments to replace the arguments specified for the invocation of the kernel image.
	read-only	Boot Linux kernel as read-only (system startup remounts as read/write).
	read-write	Mount the root system as read/write.

If you are booting an operating system from a location other than the first hard disk, you need to include a loader line for the **chain.b** file in its stanza.

```
loader=/boot/chain.b
```

Linuxconf and Webmin

Linuxconf and Webmin are two popular comprehensive administration tools that let you perform almost all your administrative tasks. This has the advantage of having to use only one software program to manage your system. Neither are currently installed with Red Hat, though you can download and install Red Hat versions for their Web sites.

Linuxconf is a comprehensive configuration tool for almost all your administrative tasks, including user and file system management, as well as network services. Linuxconf is designed to work on any Linux distribution and is currently compatible with Caldera, Red Hat, SuSe, Slackware, and Debian. Both compressed archive and RPM versions of the software are provided. You can download the current version from the Linuxconf Web site at **www.solucorp.qc.ca/linuxconf**. Here, you can also find documentation and links to any added packages.

Webmin is a Web-based interface for Unix and Linux system administration tasks. You can use any Web browser that supports tables and forms to access Webmin and perform extensive administrative operations. Webmin itself is composed of a simple Web server with CGI Perl programs with which you can directly update system configuration files such as **/etc/passwd**. Webmin is available free of charge under the BSD license. You can find out more about Webmin at **www.webmin.com**.

Webmin is an extensible application designed to be enhanced by modules that any users can write and plug into their versions. A number of standard modules are provided with Webmin that cover areas such as user account setup, LILO modification, and server configuration. Numerous third-party modules are under development. You can access them at **www.thirdpartymodules.com/webmin**.

Grand Unified Bootloader (GRUB)

The Grand Unified Bootloader (GRUB) is a multiboot boot loader that operates similar to LILO. With Red Hat 7.2, it is now the default boot loader used on Red Hat systems. GRUB offers extensive compatibility with a variety of operating system. Users can select operating systems to run from a menu interface displayed when a system boots up. Use arrow keys to move to an entry and press ENTER. Press **e** to edit a command, letting you change kernel arguments or specify a different kernel. The **c** command places you in a command line interface, similar to LILO. Provided your system bios supports very large drives, GRUB can boot from anywhere on them. Use **info grub** to list detailed documentation.

GRUB configuration is held in the **/etc/grub.conf** file. You only need to make your entries, and GRUB will automatically read them when you reboot. There are several options you can set such as the timeout period and the background image to use. You can specify a system to boot by creating a title entry for it, beginning with the term title. You then have to specify where the operating system kernel or program is located, which hard drive and what partition on that hard drive. This information is listed in parenthesis following the root option. Numbering starts from 0, not 1, and hard drives are indicated with an hd prefix. So root(hd0,2) references the first hard drive (hda) and the third partition on that hard drive (hda3). For Linux systems, you will also have to use the kernel option to indicate the kernel program to run, using the full path name and any options the kernel may need. The Ram disk is indicated by the initrd option.

```
title Red Hat Linux (2.4.7-10)
      root (hd0,2)
      kernel /boot/vmlinuz-2.4.7-10 ro root=/dev/hda3
      initrd /boot/initrd-2.4.7-10.img
```

For another operating system such as Windows, you would use the root option to specify where Windows is installed. Use the **imakeative** and **chainloader+1** options to allow GRUB to access it. Windows systems will all want to boot from the first partition on the first disk. This becomes a problem if you want to install several versions of Windows on different partitions or install Windows on a partition other than the first one. GRUB lets you work around this by letting you hide other partitions inline and then unhiding the one you want, making it appear to be the first partition. Use the **rootnoverify** command to allow the system to boot. In this example, the first partition is hidden, and the second is unhidden. This assumes there is a Windows system on the second partition on the first hard drive (hd0,1). Now that the first partition is hidden, the second one appears as the first partition.

```
hide (hd0,0)
unhide (hd0,1)
rootnoverify (hd0,1)
```

A sample **grub.conf** file follows with entries for both Linux and Windows. Notice that kernel parameters are listed in the kernel option as arguments to the kernel.

/etc/grub.conf

```
# grub.conf generated by anaconda
#
#boot=/dev/hda
default=0
timeout=30
splashimage=(hd0,2)/boot/grub/splash.xpm.gz
title Red Hat Linux (2.4.7-10)
        root (hd0,2)
        kernel /boot/vmlinuz-2.4.7-10 ro root=/dev/hda3 hdc=ide-scsi
        initrd /boot/initrd-2.4.7-10.img
title Windows XP
        root (hd0,0)
        imakeactive
        chainloader +1
```

Chapter 30: Managing Users

Overview

Linux is designed to serve many users at the same time, as well as provide an interface between the users and the computer with its storage media, such as hard disks and tapes. Users have their own shells through which they interact with the operating system. As a system administrator, you can manage user logins on your system. You can add or remove users, as well as add and remove groups. You also have access to system initialization files you can use to configure all user shells. And you have control over the default initialization files copied into an account when it is first created. With them, you can decide how accounts should initially be configured.

Note Every file is owned by a user, even those that are used by services like FTP. In such a case, a special user is created for just that service. For example, for FTP there will be a user named **ftp** that will own FTP files.

You can find out which users are currently logged in with the **who** command. Add the **-u** option to display information about each connected user, such as from where they have logged in and how long they have been inactive. The command displays the login name, the login port, the date and time of login, the length of inactivity (if still active), and the process ID for the login shell. For example:

```
# who -u
root console Oct 12 10:34 . 1219
valerie tty1 Oct 12 22:18 10 1492
```

Any utility to add a user, such as Red Hat User Manager, makes use of certain default files, called configuration files, and directories to set up the new account. A set of path names is used to locate these default files or to know where to create certain user directories. For example, **/etc/skel** holds initialization files for a new user. **/etc/password** is the file that holds user passwords. A new user's home directory is placed in the **/home** directory. Certain files provide added security such as **/etc/s**hadow, which encrypts password entries. A list of the pathnames follows:

Directory	**Description**
/home	Location of the user's own home directory.
/etc/skel	Holds the default initialization files for the login shell, such as .bash_profile, .bashrc, and .bash_logout. Includes manu user setup directories and files such as .kde for KDE and Desktop for Gnome.
/etc/shells	Holds the login shells, such as BASH or TCSH.
/etc/passwd	Holds the password for a user.
/etc/group	Holds the group to which the user belongs.
/etc/shadow	Encrypted password file.
/etc/gshadow	Encrypted password file for groups.
/etc/login.defs	Default login definitions for users.

The Password Files

When you add a user, an entry for that user is made in the **/etc/passwd** file, commonly known as the *password file.* Each entry takes up one line that has several fields separated by colons. The fields are as follows:

Field	**Description**
Username	Login name of the user
Password	Encrypted password for the user's account
User ID	Unique number assigned by the system
Group ID	Number used to identify the group to which the user belongs
Comment	Any user information, such as the user's full name
Home directory	The user's home directory
Login shell	Shell to run when the user logs in; this is the default shell, usually /bin/bash

The following is an example of a **/etc/passwd** entry. The entry for **chris** has an ***** in its Password field, indicating a password has not yet been created for this user. For such entries, you must use **passwd** to create a password. Notice also, user IDs in this particular system start at 500 and increment by one.

```
dylan:YOTPd3Pyy9hAc:500:500:User:/home/dylan:/bin/bash
chris:*:501:501:User:/home/chris:/bin/bash
```

The **/etc/passwd** file is a text file you can edit using a text editor. You can change fields in entries and even add new entries. The only field you cannot effectively change is the password, which must be encrypted. To change the Password field, you should always use the **passwd** command.

Although you can make entries directly to the **/etc/passwd** file, an easier and safer way is to use the userconf, adduser, and useradd utilities. These programs not only make entries in the

/etc/passwd file, but they also create the home directory for the user and install initialization files in the user's home directory.

The **/etc/passwd** file is a simple text file and is vulnerable to security breaches. If anyone gains access to the **/etc/password** file, they might be able to decipher or brute force crack the passwords. On most Linux distributions , the shadow suite of applications implements a greater level of security. These include versions of useradd**,** groupadd**,** and their corresponding update and delete programs. Most other user configuration tools support shadow security measures. With shadow security, passwords are no longer kept in the **/etc/password** file. Instead, passwords are kept in a separate file called **/etc/shadow** and are heavily encrypted. Access is restricted to the root user. A corresponding password file, called **/etc/gshadow,** is also maintained for groups that require passwords. As part of the standard installation for Red Hat, shadow passwords were implemented by default. You can manually specify whether you want to use shadow passwords with the authconfig tool in the Text Mode Setup utility.

Note authconfig is a text mode tool that you can use to enable and configure various authentication tools such as NIS and LDAP servers, as well as enabling shadow passwords, LDAP, and Kerberos authentication. On Red Hat, authconfig will generate the **/etc/pam.d/system-auth** configuration file use for system services.

Managing User Environments: /etc/skel and /etc/login.defs

Each time a user logs in, two profile scripts are executed. A system profile script is the same for every user, and each user has the **.bash_profile** script in his or her home directory. The system profile script is located in the **/etc** directory and named **profile** with no preceding period. As a superuser, you can edit the profile script and put in any commands you want executed for each user when he or she logs in. For example, you may want to define a default path for commands, in case the user has not done so. Or, you may want to notify the user of recent system news or account charges.

When you first add a user to the system, you must provide the user with a skeleton version of their login, shell, and logout initialization files. For the BASH shell, this would be a **.bash_profile .bashrc** and **.bash_logout** files. The **useradd** command and other user management tools like User Manager add this file automatically, copying any files in the directory **/etc/skel** to the user's new home directory. The **/etc/skel** directory contains a skeleton initialization file for **.bash_profile, .bashrc**, and **.bash_logout** files or, if you are using the TCSH shell as your login shell, **.login**, **.tcshrc,** and **.logout** files. The **/etc/skel** directory also contains default files and directories for your desktops. These include a **.screenrc** file for the X Window System, a **.kde** directory for the KDE desktop, and a **Desktop** directory that contains default configuration files for the Gnome desktop.

As a superuser, you can configure the **.bash_profile** or **.bashrc** files in the **/etc/skel** any way you want. Usually, basic system variable assignments are included that define pathnames for commands and command aliases. On Red Hat, the **PATH** and **BAS**H_ENV variables are defined in **.bash_profile**. Once users have their own **.bash_profile** or **.bashrc** files, they can redefine variables or add new commands as they choose. The **.bashrc** file will also run the **/etc/bashrc** to implement any global definitions such as the **PS1** and **TERM** variables. The **/etc/bashrc** file also executes any specialized initialization file in the **/etc/profile.d** directory such as those used for KDE and Gnome. The **.bash_ profile** file runs the **.bashrc** file, and through it, the **/etc/bashrc** file, implementing global definitions.

System-wide values used by user and group creation utilities like useradd and usergroup are kept in the **/etc/login.defs** file. Here you will find the range of possible user and group IDs listed. UID_MIN holds the minimum number, and UID_MAX the maximum number for user IDs. Various password options control password controls- such as PASS_MIN_LEN, which determines the minimum number of characters allowable in a password. Options such as CREATE_HOME can be set to tell useradd to create home directories for new accounts by default.

login.defs

```
# *REQUIRED*
# Directory where mailboxes reside, _or_ name of file, relative to the
# home directory. If you _do_ define both, MAIL_DIR takes precedence.
# QMAIL_DIR is for Qmail
#
#QMAIL_DIR Maildir
MAIL_DIR /var/spool/mail
#MAIL_FILE .mail

# Password aging controls:
# PASS_MAX_DAYS Maximum number of days a password may be used.
# PASS_MIN_DAYS Minimum number of days allowed between password changes.
# PASS_MIN_LEN Minimum acceptable password length.
# PASS_WARN_AGE Number of days warning given before a password expires.
PASS_MAX_DAYS 99999
PASS_MIN_DAYS 0
PASS_MIN_LEN 5
PASS_WARN_AGE 7

# Min/max values for automatic uid selection in useradd
UID_MIN 500
UID_MAX 60000

# Min/max values for automatic gid selection in groupadd
GID_MIN 500
GID_MAX 60000

# If defined, this command is run when removing a user.
#USERDEL_CMD /usr/sbin/userdel_local

# If useradd should create home directories for users by default
# On RH systems, we do.
CREATE_HOME yes
```

Login Access

You can control user login access to your system with the **/etc/login.access** file. The file consists of entries listing users, whether they are allowed access, and from where they can access the system. A record in this file consists of three colon-delimited fields: a plus (+) or minus (**-**) sign indicating whether users are allowed access, user login names allowed access, and the remote system (host) or terminal (tty device) from which they are trying to log in. The following enables the user **dylan** to access the system from the **rabbit.mytrek.com** remote system:

```
+:dylan:rabbit.mytrek.com
```

You can list more than one user or location. You can also use the **ALL** option in place of either users or locations to allow access by all users and locations. The **ALL** option can be qualified with the **EXCEPT** option to allow access by all users except certain specified ones. The following entry allows any user to log into the system using the console, except for the users **larisa** and **aleina**:

```
+:ALL EXCEPT larisa aleina:console
```

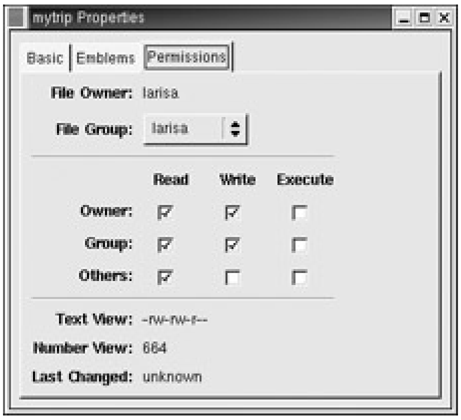
Other access control files are used to control access for specific services such as the **hosts.deny** and **hosts.allows** files used with **tcpd** daemon for xinetd-supported servers.

Controlling Access to Directories and Files

Recall from Chapter 12 that you can control access to your files and directories by user, group, or others. This is a capability given to any user for their own files and directories. Access in each of these categories can be controlled according to write, read, and execute permissions. Write access lets users modify a file, read access lets them display it, and execute access (used for programs) lets them run it. For directories, write access lets them delete it, read access will list its contents, and execute access will let users change to that directory. You could allow anyone on the system to read one of your files by assigning a read access to its other permission. Chapter 12 describes how you can use the **chmod** command to set permissions, using the **u** for user, **g** for group, and **o** for other, as well as **r** for read, **w** for write, and **x** for execute. The following command provides read and write group permissions for the file **ourdraft1**:

```
chmod g+rx ourdraft1
```

On Gnome, you can set a directory or file permission using its Permissions panel in its Properties window (see Figure 30-1). Right-click on the file or directory entry in the file manager window and select Properties. Then select the Permissions panel. Here you will find a table of boxes with columns for Read, Write, and Execute along with rows for Owner, Group, and Other. Check the appropriate box for the permission you want. Normally, the Read and Write boxes for user permission will already be set. You can specify the group you want access provided to from the Group drop-down menu. This displays the groups a user belongs to.



File Owner: larisa

File Group: larisa

	Read	Write	Execute
Owner:	☑	☑	□
Group:	☑	☑	□
Others:	☑	□	□

Text View: -rw-rw-r--

Number View: 664

Last Changed: unknown

Figure 30-1: File and directory permissions on Gnome

In addition to the read/write/execute permissions, you can manually set ownership permissions for executable programs. Normally, the user who runs a program owns it while it is running, even though the program file itself may be owned by another user. The set user ID permission allows the original owner of the program to own it always, even while another user is running the program. For example, most software on the system is owned by the root user, but is run by ordinary users. Some such software may have to modify files owned by the root. In this case, the ordinary user would need to run that program with the root retaining ownership so the program could have the permissions to change those root-owned files. The group ID permission works the same way, except for groups. Programs owned by a group retain ownership even when run by users from another group. The program can then change the owner group's files.

To add both the user ID and group ID permissions to a file, you use the **s** option. The following example adds the user ID permission to the **pppd** program, which is owned by the root user. When an ordinary user runs **pppd**, the root user retains ownership, allowing the **pppd** program to change root-owned files.

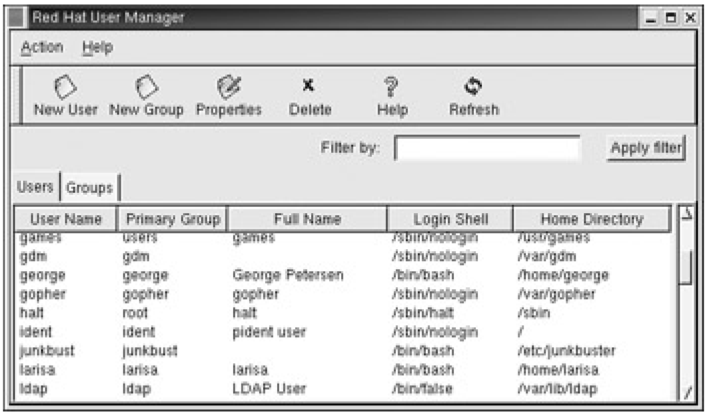
```
# chmod +s /usr/sbin/pppd
```

Note Where a program is owned by the root, setting the user ID permission will give the user the ability to execute the program with root permissions. This can be a serious security risk for any program that could effect changes-such as **rm**, which removes files.

Red Hat User Manger

Most administration tools let you easily add, remove, or change users. For Red Hat distributions, it is recommended that you use the Red Hat User Manager to manage user accounts. You can access the Red Hat User Manager from the System Settings window in the Start Here window. It will be labeled User Manager. You can also access it from the Gnome System menu.

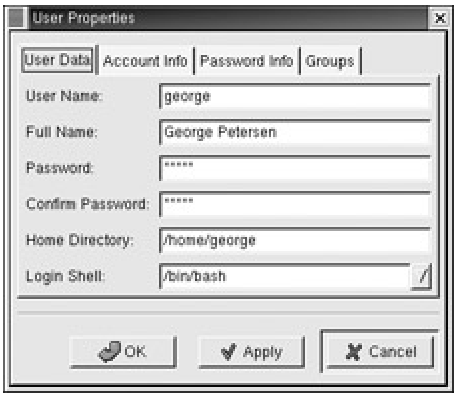
The User Manger window will display panels for listing both users and groups (see Figure 30-2). You use the User Manager to manage your groups, as well as users. Click the appropriate tab to display either users or groups. Within the User and Group panels, field labels are displayed at the top for User Name, Group, the user's Full Name, Login Shell, and Home Directory. A button bar will list various tasks you can perform, including creating new users or groups, editing current ones (Properties), or deleting a selected user or group. The number of users and groups on a system can be extensive. The User Manager provides an easy-to-use search tool. In the box labeled Filter By, you can enter a search string. Then, when you click the Apply Filter button, only those matching users or groups are listed.



User Name	Primary Group	Full Name	Login Shell	Home Directory
games	users	games	/sbin/nologin	/usr/games
gdm	gdm		/sbin/nologin	/var/gdm
george	george	George Petersen	/bin/bash	/home/george
gopher	gopher	gopher	/sbin/nologin	/var/gopher
halt	root	halt	/sbin/halt	/sbin
ident	ident	pident user	/sbin/nologin	/
junkbust	junkbust		/bin/bash	/etc/junkbuster
larisa	larisa	larisa	/bin/bash	/home/larisa
ldap	ldap	LDAP User	/bin/false	/var/lib/ldap

Figure 30-2: Linuxconf user accounts

To create a new user, click the New User button. This opens a window with entries for the user name, password, login shell, along with options to create a home directory and a new group for that user. Once you have created a user, you can edit its properties to add or change features. Select the user's entry and click the Properties button. This displays a window with tabbed panels for User Data, Account Info, Password Info, and Groups (see Figure 30-3). You can change basic features such as the password and login shell in the User Data panel. Account Info lets you lock an account and set an expiration date for it. Password Info will let you set password expiration limits to force a user to change the password or to render the account inactive after a certain time. On the Groups panel, you can select the groups that the user belongs to, adding or removing group membership.



User Name: george

Full Name: George Petersen

Password: *****

Confirm Password: *****

Home Directory: /home/george

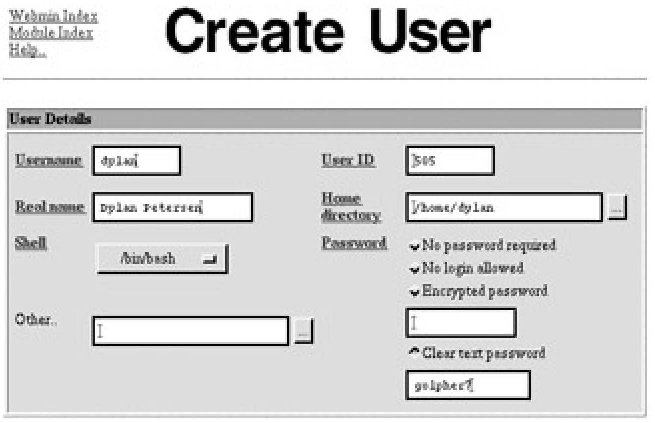
Login Shell: /bin/bash

Figure 30-3: Linuxconf user information

Note You can also manage users with Linuxconf, as was done in older versions of Red Hat.

Managing Users with Webmin

To manage users with Webmin, select the Users and Groups entry in the System panel. This displays a listing of all your users and groups. To create a new user, click on the Create a New User link below the list of users. The Create User page displays segments for User Details, Password Options, and Group Memberships. For User Details, you enter the Username (login name), the Real Name, the Shell (login shell), the Home Directory, and the Password (See Figure 30-4). For the shell entry, you can choose the shell you want from a drop-down menu. For the home directory, you can type in the directory name or click on the button next to the entry box labeled with three dots. This opens a file manager interface where you can click to the directory and file you want.



Username: dylan

User ID: 505

Real name: Dylan Petersen

Home directory: /home/dylan

Shell: /bin/bash

Password:
○ No password required
○ No login allowed
○ Encrypted password
● Clear text password: gelpher7

Other..

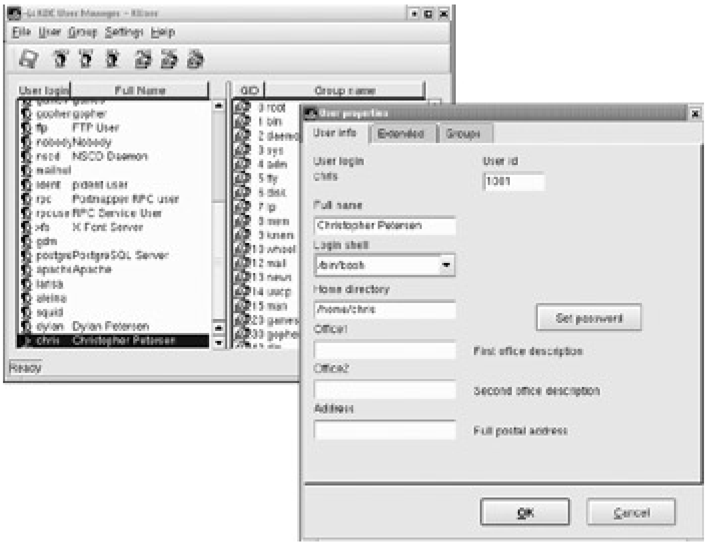
Figure 30-4: Webmin user details

For the password options, you can specify the period when the user needs to change the password and even set an expiration date. For group membership, you can select the primary (default) group (usually **users**) and add any secondary groups.

Once a group is created, an entry for it will appear in the Users and Groups page. The user name will appear as a link. To make any changes or to delete the group, simply click on its name. The Edit User page will appear with entries for changing the password, changing secondary groups, or deleting the user.

Managing Users with kuser

The K Desktop also provides a simple user management utility called kuser that works much like Linuxconf (see Figure 30-5). You can use it to manage both users and groups. The window is divided into two panes: one for users and the other for groups. Add, Edit, and Delete icons easily enable you to add new users, change their configuration, or remove them. When you add a new user, a new window opens with entries such as the shell and home directory. To add the password, click Password and enter the password in the window displayed.



User login	Full Name
gopher	gopher
ftp	FTP User
nobody	Nobody
nscd	NSCD Daemon
mailnull	
ident	pident user
rpc	Portmapper RPC user
rpcuser	RPC Service User
xfs	X Font Server
gdm	
postgres	PostgreSQL Server
apache	Apache
lamsa	
aleina	
squid	
dylan	Dylan Petersen
chris	Christopher Petersen

GID	Group name
0	root
1	bin
2	daemon
3	sys
4	adm
5	tty
6	disk
7	lp
8	mem
9	kmem
10	wheel
12	mail
13	news
14	uucp
15	man
20	games
30	gopher

User login: chris
User id: 1001
Full name: Christopher Petersen
Login shell: /bin/bash
Home directory: /home/chris
Office1:
Office2:
Address:

Figure 30-5: kuser

Adding and Removing Users with useradd, usermod, and userdel

Red Hat also provides the **useradd**, **usermod**, and **userdel** commands to manage user accounts. All these commands take in all their information as options on the command line. If an option is not specified, they use predetermined default values. With the **useradd** command, you enter values as options on the command line, such as the name of a user to create a user account. It then creates a new login and directory of that name using all the default features for a new account.

```
# useradd chris
```

The useradd utility will first check the **/etc/login.defs** file for default values for creating a new account. For those defaults not defined in **/etc/login.defs** file, useradd supplies its own. You can display these defaults using the **useradd** command with the **-D** option. The default values include the group name, the user ID, the home directory, the **skel** directory, and the login shell. Values the user enters on the command line will override corresponding defaults. The group name is the name of the group in which the new account is placed. By default, this is *other,* which means the new account belongs to no group. The user ID is a number identifying the user account. This starts at 500 on Red Hat with the first account and increments automatically for each new account. The **skel** directory is the system directory that holds copies of initialization files. These initialization files are copied into the user's new home directory when it is created. The login shell is the pathname for the particular shell the user plans to use.

The **useradd** command has options that correspond to each default value. Table 30-1 holds a list of all the options you can use with the **useradd** command. You can use specific values in place of any of these defaults when creating a particular account. The login is inaccessible until you do. In the next example, the group name for the **chris** account is set to **intro1** and the user ID is set to 578:

```
# useradd chris -g intro1 -u 578
```

Table 30-1: User and Group Management Commands

Command	**Description**

Table 30-1: User and Group Management Commands

Command	**Description**
useradd *username options*	Adds new users to the system.
usermod *username options*	Modifies a user's features.
userdel -r *username*	Removes a user from the system.
useradd, **usermod**	
Options	
-c *str*	Adds a comment to the user's entry in the system password file: **/etc/passwd**.
-d *dir*	Sets the home directory of the new user.
-D	Displays defaults for all settings. Can also be used to reset default settings for the home directory(**-b**), group (**-g**), shell(**-s**), expiration date (**-e**), and password expirations (**-f**).
-e *mm/dd/yy*	Set an expiration date for the account (None, by default). Specify by month/day/year.
-f *days*	Sets the number of days an account remains active after its password expires.
-g *group*	Sets a group.
-G *group*	Sets additional groups.
-m	Create user's home directory if it does not exist.
-m -k *skl-dir*	Sets the skeleton directory that holds skeleton files, such as **.profile** files, which are copied to the user's home directory automatically when it is created; the default is **/etc/skel**.
-M	Does not create user's home directory.
-n	Turns off the Red Hat-specific default procedure whereby a new group is created with the same name as a new account's user name. For example, a new user, **dylan**, would have as its group, **dylan**.
-p *password*	Supply an encrypted password (crypt or MD5). With no argument, the account is immediately disabled.
-r	A Red Hat-specific option that will create a system account (one whose user ID is lower than the minimum set in **logon.defs**). No home directory is created unless specified by **-m**.
-s *shell*	Sets the login shell of the new user. This is the **/bin/bash** by default, the BASH shell.
-u *userid*	Sets the user ID of the new user; the default is the increment of the highest number used so far.
Group Management Commands	

Table 30-1: User and Group Management Commands

Command	Description
groupadd *groupname options*	Creates a new group.
Group Management Commands	
groupdel *groupname options*	Removes a group.
groupmod *groupname options*	Modifies a group.
useradd, **usermod**	
Options	
-g *gid*	Change a group ID.
-n *groupname*	Change a group name.
-f	In Red Hat, detects if group already exists.
-r	In Red Hat, creates a system group, one lower than the group minimum specified in **login.defs**.

Once you add a new user login, you need to give the new login a password. Password entries are placed in the **/etc/passwd** and **/etc/shadow** files. Use the **passwd** command to create a new password for the user, as shown here. The password you enter will not appear on your screen. You will be prompted to repeat the password. A message will then be issued indicating that the password was successfully changed.

```
# passwd chris
Changing password for user chris
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
#
```

The **usermod** command enables you to change the values for any of these features. You can change the home directory or the user ID. You can even change the user name for the account.

When you want to remove a user from the system, you can use the **userdel** command to delete the user's login. In the next example, the user **chris** is removed from the system:

```
# userdel -r chris
```

Note You can also add a new user to the system with the adduser utility. The Red Hat version of adduser takes as its argument the user name for the account you are creating. adduser has options similar to useradd.

Managing Groups

You can manage groups using either shell commands or window utilities like Linuxconf. The system file that holds group entries is called **/etc/group**. The file consists of group records, with one record per line and its fields separated by colons. A group record has four fields: a group name, a password, its ID, and the users who are part of this group. The Password field can be left blank. The fields for a group record are as follows:

Group name	Name of the group; must be unique

Group name	**Name of the group; must be unique**
Password	Usually an asterisk to allow anyone to join the group; a password can be added to control access
Group ID	Number assigned by the system to identify this group
Users	List of users that belong to the group

Here is an example of an entry in an **/etc/group** file. The group is called **engines**, there is no password, the group ID is 100, and the users who are part of this group are **chris**, **robert**, **valerie**, and **aleina**.

```
engines::100:chris,robert,valerie,aleina
```

As in the case of the **/etc/passwd** file, you can edit the **/etc/group** file directly using a text editor, unless you have implemented shadow security. Instead of using either Linuxconf or **groupdel**, you could simply delete the entry for that group in the /etc/group file. This can be risky, however, if you make accidental changes. If you have implemented shadow security measures, then group entries are kept in an encrypted **/etc/gshadow** file. You cannot edit this file. Changes can only be made through a utility like **groupmod**.

As for users, you can also create a home directory for a group. Several administration utilities like Linuxconf support this feature. To do so manually, you simply create a directory for the group in the **/home** directory and change its group to that of the group, along with allowing access by any member of the group. The following example creates a directory called **engines** and changes its group to that of the **engines** group:

```
mkdir /home/engines
chgrp engines /home/engines
```

Then the read, write, and execute permissions for the group level should be set:

```
chmod g+rwx /home/engines
```

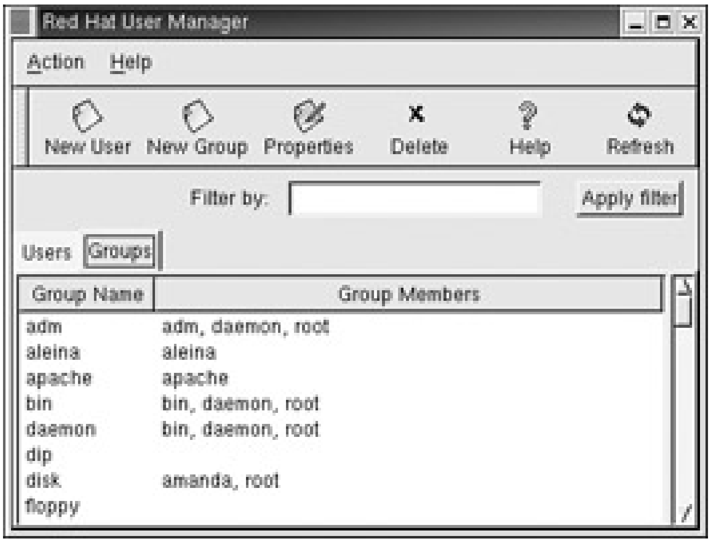
Any member of the **engines** group can now access the **/home/engines** directory and any shared files placed therein. This directory becomes a shared directory for the group. You can, in fact, use the same procedure to make other shared directories at any location on the file system.

Files within the shared directory should also have their permissions set to allow access by other users in the group. When a user places a file in a shared directory, permissions on that file need to be changed to allow other members of the group to access it. A read permission will let others display it, write lets them change it, and execute lets then run it (used for scripts and programs). The following example first changes the group for the **mymodel** file to **engines**. Then it copies the **mymodel** file to the **/home/engines** directory and then sets the group read and write permission for the **engines** group.

```
$ chgrp engines mymodel
$ cp mymodel /home/engines
$ chmod g+rw /home/engines/mymodel
```

Managing Groups with the Red Hat User Manager

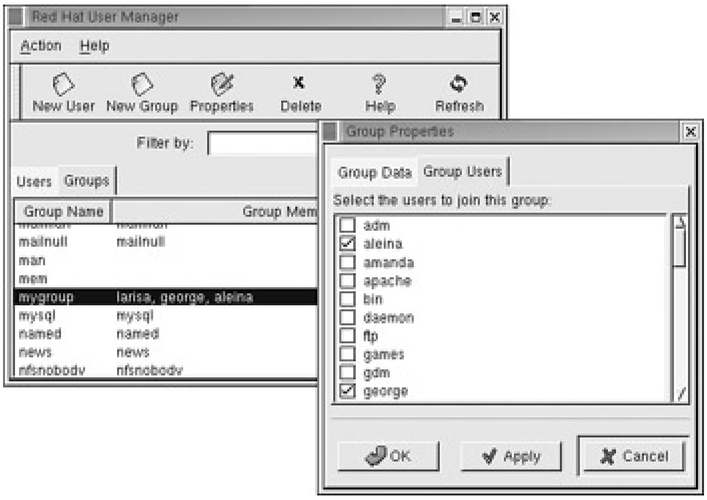
You can add, remove, and modify any groups easily with the Red Hat User Manager. First access the Red Hat User Manager by clicking the User Manager icon in the System Settings window, listed on the Start Here window. Then click on the tabbed panel labeled Groups in the Red Hat User Manager window. All your current groups will be listed. There will be two fields for each entry, the Group Name and the Group Members (see Figure 30-6).



Group Name	Group Members
adm	adm, daemon, root
aleina	aleina
apache	apache
bin	bin, daemon, root
daemon	bin, daemon, root
dip	
disk	amanda, root
floppy	

Figure 30-6: The Linuxconf Group panel

To add a group, just click the New Group button. This opens a small window where you can enter the group name. When you click the Refresh button, the new group will be listed in the User Manager's Group listing. To add users as members of the group, select the group's entry and click the Properties button. This opens a window with tabbed panels for both the Group Data and the Group Users. The Groups Users panel will list all current users with checkboxes beside their name (see Figure 30-7). Click the checkboxes for the user you want to add to the group. Should you want to remove a user as a member of the group, click its checkbox to remove its checkmark. Click Apply to apply your changes.



Group Name	Group Mem
mailnull	mailnull
man	
mem	
mygroup	larisa, george, aleina
mysql	mysql
named	named
news	news
nfsnobody	nfsnobody

Select the users to join this group:

- □ adm
- ☑ aleina
- □ amanda
- □ apache
- □ bin
- □ daemon
- □ ftp
- □ games
- □ gdm
- ☑ george

Figure 30-7: Linuxconf Group Specification panel

Should you want to remove a group, just select its entry in the User Manger's Group panel and then click the Delete button.

Note You can also mange groups with Linuxconf.

Be sure to check the permissions and group ownership for any group directory that you create. The group permissions need to be set to read, write, and execute for full access. The read permission will let users list files in the directory, the write will let users add and remove files, and the execute permission lets users change to that directory. As the root user, you can use **chmod** to set these permissions:

```
chmod 775 /home/golphpros
```

Then, to change the group ownership, you use **chgrp**, as shown here:

```
chgrp golphpros /home/golphpros
```

In Gnome, you can check and change permission with the Gnome file manager by right-clicking on the directory name and selecting Properties. Then select the Permissions panel. For full access, make sure the row of Group Permission buttons is selected. Also, select the group from the list of groups in the Group drop-down menu-in this example, **golphpros**.

Managing Groups Using Webmin

To manage groups in Webmin, you select the Users and Groups entry in the Systems panel to display the Users and Groups page. Move down to the group listings. Here, groups are listed with their names functioning as links to an Edit Group page for that group. In the Edit Group page, you can add a password, change the ID, or add new users to the group. To create a group, click on the Create New Group link following the Group list on the Users and Groups page. This opens a Create Group page where you can add a group name, add a password, and add users to the group. The group ID is provided for you, which you can then change if you wish. When you add users to a group, a new window opens up with two panes, one listing all users and the other for the users you have selected for this group. Each user name is a link. Clicking on a user name in the All Users lists adds it to the Selected Users list. Clicking on a name in the Selected Users list removes it from that list. When you click the OK button, the names you have selected will appear in the Members box on the Create Group page.

Managing Groups Using groupadd, groupmod, and groupdel

You can also manage groups with the **groupadd**, **groupmod**, and **groupdel** commands. With the **groupadd** command, you can create new groups. When you add a group to the system, the system places the group's name in the **/etc/group** file and gives it a group ID number. If shadow security is in place, then changes are made to the **/etc/gshadow** file. The **groupadd** command only creates the group category. Users are individually added to the group. In the next example, the **groupadd** command creates the **engines** group:

```
# groupadd engines
```

You can delete a group with the **groupdel** command. In the next example, the **engines** group is deleted:

```
# groupdel engines
```

You can change the name of a group or its ID using the **groupmod** command. Enter **groupmod** -g with the new ID number and the group name. To change the name of a group, you use the **-n** option. Enter **groupmod -n** with the new name of the group, followed by the current name. In the next example, the **engines** group has its name changed to **trains**:

```
# groupmod -n trains engines
```

Disk Quotas

With disk quotas, you can control how much disk space a particular user makes use of on your system. On your Linux system, unused disk space is held as a common resource that each user can access as they need it. As a user creates more files, they take the space they need from the pool of available disk space. In this sense, all the users are sharing this one resource of unused disk space. However, if one user were to use up all the remaining disk space, then none of the other users would be able to create files or even run programs. To counter this problem, you can create disk quotas on particular users, limiting the amount of available disk space they can use.

Note On Linuxconf, you can enable disk quotas and set general controls. To enable quotas for a disk, first select the disk in the Access Local Drive panel under File Systems in the Config panel. Then click the disk's Options panel and click on the User Quota Enabled and Group Quota Enabled check boxes. Once you accept and activate the changes, you can then access the Set Quota Defaults panel under File systems. Here, you can set user and group defaults for disk space and file limits as well as grace periods.

Quotas are enabled using the quotacheck and quotaon programs. On Red Hat, they are executed in the **/etc/rc.d/rc.sysinit** script, which is run whenever you start up your system. Each partition then needs to be mounted with the quota options, **usrquota** or **grpquota**. **usrquota** enables quota controls and users, and **grpquota** works for groups. These options are usually placed in the mount entry in the **/etc/fstab** file for a particular partition. For example, to mount the **/dev/hda6** hard disk partition mounted to the **/home** directory with support for user and group quotas, you would require a mount entry like the following:

```
/dev/hda6 /home ext2 defaults,usrquota,grpquota 1 1
```

You also need to create **quota.user** and **quota.group** files for each partition for which you enable quotas. These are the quota databases used to hold the quota information for each user and group. You can create these files by running the **quotacheck** command with the **-a** option or the device name of the file system where you want to enable quotas.

The limit you set for a quota can be hard or soft. A hard limit will deny a user the ability to exceed his or her quota, whereas a soft limit will just issue a warning. For the soft limit, you can designate a grace period during which time the user has the chance to reduce his or her disk space below the limit. If the disk space still exceeds the limit after the grace period expires, then the user can be denied access to his or her account.

You can set disk quotas using the **edquota** command or the User Account panels in Linuxconf. On Linuxconf, select a user from the User Accounts listing and then select that user's Disk Quota panel. Here, you can enter the hard and soft limits along with the grace period. You can also select a default for these limits. The default is set using the Set Quota Defaults panel in the File System menu. There are entries for both group and user soft and hard defaults. Linuxconf will allow you to place limits on both the disk size and the number of files.

The **edquota** command is run from the command line. With it you can access the quota record for a particular user and group, which is maintained in the disk quota database. You can also set default quotas that will be applied to any user or group on the file system for which quotas have not been set. **edquota** will open the record in your default editor, and you can use your editor to make any changes. To open the record for a particular user, use the **-u** option and the user name as an argument for **edquota** (see Table 30-2). The following example opens the disk quota record for the user **larisa**:

```
edquota -u larisa
```

Table 30-2: The options edquota and quota

edquota	**Description**
-u	Edit the user quota. This is the default.
-g	Edit the group quota.
-p	Duplicate the quotas of the prototypical user specified for each user specified. This is the normal mechanism used to initialize quotas for groups of users.
-t	Edit the soft time limits for each file system.
quota	Description
-g	Print group quotas for the group of which the user is a member.
-u	Print the user's quota.
-v	Will display quotas on file systems where no storage is allocated.
-q	Print information on file systems where usage is over quota.

The quota record begins with the hard disk device name and the blocks of memory and inodes in use. The Limits segments have parameters for soft and hard limits. If these entries are 0, there are no limits in place. You can set both hard and soft limits, using the hard limit as a firm restriction. Blocks in Linux are currently about 1,000 bytes. The inodes are used by files to hold information about the memory blocks making up a file. To set the time limit for a soft limit, you use the **edquota** command with the **-t** option. The following example displays the quota record for **larisa**:

```
Quotas for user larisa:
/dev/hda3: blocks in use: 9000, limits (soft = 40000, hard = 60000)
 inodes in use: 321, limits (soft = 0, hard = 0)
```

These records are maintained in the quota database for that partition. Each partition that has quotas enabled will have its own quota database. You can check the validity of your quota database with the **quotacheck** command. You can turn quotas on an off using the **quotaon** and **quotaoff** commands. When you start up your system, **quotacheck** is run to check the quota databases and then **quotaon** is run to turn on quotas.

As the system administrator, you can use the **repquota** command to generate a summary of memory usage, checking to see what users are approaching or exceeding quota limits. Individual users can use the **quota** command to check their memory use and how much disk space they have left in their quota (see Table 30-2).

Lightweight Directory Access Protocol

The Lightweight Directory Access Protocol (LDAP) is designed to implement network-accessible directories of users, providing information about them such as their e-mail address or phone numbers. Such directories can also be used for authentication purposes, identifying that a certain user belongs to a specified network. You can find out more information on LDAP at **www.ldapman.org**. You can think of an LDAP directory as an Internet-accessible phone book, where anyone can look you up to find your e-mail address or other information. In fact, it may be more accurate to refer to such directories as databases. They are databases of user information, accessible over networks like the Internet. Normally, the users on a local network are spread across several different systems. Ordinarily, to obtain information about a user, you would have to know what system the user is on and then query that system. With LDAP, user information for all users on a network is kept in the LDAP server. You only have to query the network's LDAP server to obtain information about a user. For example, Sendmail can use LDAP to look up user addresses.

With LDAP, you can also more carefully control the kind of information given and to whom. Using a PAM module (pam_ldap), LDAP can perform user authentication tasks, providing centralized authentication for users. Login operations that users perform for different services such as mail POP server, system login, and Samba logins can all be carried out through LDAP using a single PAM secured user ID and password.

Note On Red Hat, you can use the GQ (Gnome) and kldap (KDE) clients to query LDAP servers.

LDAP directories are implemented as clients and servers, where you use an LDAP client to access an LDAP server that manages the LDAP database. Red Hat uses OpenLDAP, an open source version of LDAP (you can find out more about OpenLDAP at **www.openldap.org**). This package includes an LDAP server (slapd), an LDAP replication server (slurpd), an LDAP client, and tools. slurpd is used to update other LDAP servers on your network, should you have more than one. Once the LDAP server is installed, you can start, stop, and restart the LDAP server with the **ldap** startup script.

```
service ldaprestart
```

An entry in an LDAP database will consist of a name (known as a distinguished name) followed by a set of attributes and their values. For example, a name could be a user name and the attribute would be the user's e-mail address, the address being the attribute's value. Allowable attributes are determined by object class sets defined in the **/etc/openldap/schema** directory. To actually make entries in the LDAP database, you use the ldapadd and ldapmodify utilities. With ldapdelete, you can remove entries. Once you have created an LDAP database, you can then query it, through the LDAP server, with ldapsearch. You can also create a text file of LDAP entries using an LDAP Data Interchange Format (LDIF) format. Such text files can then be read in all at once to the LDAP database using the slapadd utility.

Note You can enable and designate LDAP servers with the authconfig tool. You can also use the LDAP Browser/Editor or the Gnome Directory Administrator to manage and edit LDAP directories.

All LDAP configuration files are kept in the **/etc/openldap** directory. These include **sldap.conf**, the LDAP server configuration file, and **ldap**, the LDAP clients and tools configuration file. In the **sldap.conf** file, the suffix entry should be changed to your own network's domain address. This is the network that will be serviced by the LDAP server. You will have to perform the same task for the **ldap.conf** file, to enable use of your LDAP clients and tools like ldapadd. For LDAP authentication, you will to make the same modifications for **nss_ldap** and **pam_ldap** files. To use LDAP for authentication, you will need to configure PAM to use it, as well as migrate authentication files to the LDAP format. The **/usr/share/openldap/migration** directory holds scripts you can use to translate the old files into LDAP versions.

Pluggable Authentication Modules (PAM)

Pluggable Authentication Modules (PAM) is an authentication service that lets a system determine the method of authentication to be performed for users. In a Linux system, authentication has traditionally been performed by looking up passwords. When a user logs in, the login process will look up his or her password in the password file. With PAM, users' requests for authentication are directed to PAM, which in turn will use a specified method to authenticate the user. This could be a simple password lookup or a request to an LDAP server. But it is PAM that provides authentication, not a direct password lookup by the user or application. In this respect, authentication becomes centralized and controlled by a specific service, PAM. The actual authentication procedures can be dynamically configured by the system administrator. Authentication is carried out by modules that can vary according to the kind of authentication needed. An administrator can add or replace modules by simply changing the PAM configuration files. See the PAM Web site at **www.kernel.org/pub/linux/libs/pam** for more information and a listing of PAM modules. On Red Hat, PAM modules are located in the **/lib/security** directory.

On Red Hat, PAM uses different configuration files for different services that would request authentication. Such configuration files are kept in the **/etc/pam.d** directory. For example, you have a configuration file for logging into your system (**/etc/pam. d/login**), one for the graphical login (**/etc/pam.d/gdm**), and one for access your Samba server (**/etc/pam.d/samba**). A default PAM configuration file, called **/etc/pam.d/other**, is invoked if no services file is present. On Red Hat, the **system-auth** file contains standard authentication modules for system services generated by authconfig and is invoked in many of the other configuration files.

A PAM configuration file contains a list of modules to be used for the authentication. They have the following format:

```
module-type control-flag module-path module-arguments
```

The *module-path* is the module to be run, and *module-arguments* are the parameters you want passed to that module. Though there are a few generic arguments, most modules have their own. The *module-type* refers to different groups of authentication management: account, authentication, session, and password. The account management performs account verification, checking such account aspects as whether the user has access, or whether the password has expired. Authentication (**auth**) verifies who the user is, usually through a password confirmation. Password management performs authentication updates such as password changes. Session management refers to tasks performed before a service is accessed and before it is left. These include tasks like initiating a log of a user's activity or mounting and unmounting home directories.

Note As an alternative to the **/etc/pam.d** directory, you could create one configuration file called the **/etc/pam.conf** file. Entries in this file have a service field, which refers to the application that the module is used for. If the **/etc/pam.d** directory exits, as it does in Red Hat, **/etc/pam.conf** is automatically ignored.

The *control-flag* field indicates how PAM will respond if the module fails. The control can be a simple directive or a more complicated response that can specify return codes like **open_err** with actions to take. The simple directives are **requisite**, **required**, **sufficient**, and **optional**. The **requisite** directive will end the authentication process immediately if the module fails to authenticate. The **required** directive will only end the authentication after the remaining modules are run. The **sufficient** directive indicates that success of this module is enough to provide authentication unless a previous required module has failed. The **optional** directive indicates the module's success is not needed unless it is the only authentication module for its service. If you specify return codes, you can refine the conditions for authentication failure or success. Return codes can be given values such as **die** or **ok**. The **open_err** return code could be given the action **die**, which would stop all authentication and return failure.

The **/etc/pam.d/ftpwu** configuration file for the FTP server is shown here.

ftpwu

```
#%PAM-1.0
auth       required     /lib/security/pam_listfile.so item=user
                        sense=deny file=/etc/ftpusers onerr=succeed
auth       required     /lib/security/pam_stack.so service=system-auth
auth       required     /lib/security/pam_shells.so
account    required     /lib/security/pam_stack.so service=system-auth
session    required     /lib/security/pam_stack.so service=system-auth
```

Chapter 31: Software Management

Overview

Installing or updating software packages has always been a simple process in Linux due to the widespread use of the Red Hat Package Manager. Instead of using a standard tar archive, software is packaged in a special archive for use with the Red Hat Package Manager. These archives have become known as RPMs, where *RPM* stands for *Red Hat Package Manager.* An RPM archive contains all the program files, configuration files, data files, and even documentation that constitute a software application. With one simple operation, the Red Hat Package Manager installs all these for you from an RPM software package. You can even create your own RPM packages. You can use any of several RPM window-based utilities to manage your RPM packages, installing new ones or uninstalling ones you have. These utilities provide an easy-to-use interface for managing your packages, enabling you to obtain detailed information on a package easily, including a complete listing of the files it installs. Also, as part of their administration tools, distributions like Red Hat also provide software management for packages on their CD-ROMs.

Note The Red Hat Update Agent, through the Red Hat Network, will automatically download and update any Red Hat RPM packages for you that are installed on your system and are part of the Red Hat distribution.

You can also download source code versions of applications, and then compile and install them on your system. Where this process once was complex, it has been significantly streamlined with the addition of configure scripts. Most current source code, including GNU software, is distributed with a configure script. The *configure script* automatically detects your system configuration and generates a **Makefile** with which a binary file is created compatible to your system. With three simple commands, you can compile and install complex source code on any system.

Extensive online sources exist for downloading Linux software. Sites are available for particular kinds of applications, such as Gnome and KDE, as well as for particular distributions, such as Red Hat. As you have seen, the Red Hat Network can automatically download and update software installed from RPM packages that make up the Red Hat distribution (see Chapter 4). Some sites are repositories for RPM packages, such as **rpmfind.net,** while others like **freshmeat.net** refer you to original development sites where you can download software packages. The **freshmeat.net** and **www.linuxapps.com** sites are useful for finding out about new available software. Many of the open source Linux projects can now be found at **sourceforge.net**. Here you will find detailed documentation and recent versions of software packages. For applications designed for the Gnome desktop, you can check **www.gnome.org**, you can find KDE applications at **apps.kde.com**. For particular database and office applications, you can download software packages directly from the company's Web site, such as **www.sun.com** for the StarOffice office suite and **www.oracle.com** for the Oracle database (see Chapters 22 and 23). For RPM packages , which are not part of a Red Hat distribution, you can check the **contrib** and **powertools** directories on the Red Hat FTP site (**ftp.redhat.com)**. Here, you can find Red Hat RPM packages for applications, such as ProFTPD and htDig. Table 31-1 lists several popular Linux software sites.

Table 31-1: Linux Software Sites

FTP and Web Sites	Applications
ftp.redhat.com	Software packaged in RPM packages for Red Hat. Check the **contrib** and **powertools** directories for contributed software.
freshmeat.net	Linux software, includes RPMs.
linuxapps.com	Linux software, includes RPMs.
rpmfind.net	RPM package repository.
www.sourceforge.net	Linux open source software projects.
www.gnome.org	Gnome software, includes RPMs.
apps.kde.com	KDE software, includes RPMs.
http://home.xnet.com/~blatura/linapps.shtml	Linux applications and utilities page.

Table 31-1: Linux Software Sites

FTP and Web Sites	Applications
www.filewatcher.org	Linux FTP site watcher.
www.gnu.org	GNU archive.
www.ximian.com	Ximian Gnome, office applications for Gnome.
koffice.kde.com	The KDE KOffice suite of office applications.
www.xdt.com/ar/linux-snd	Linux MIDI and sound pages.
www.linuxvideo.org	The Linux Video and DVD Project, LiViD.
www.opensound.com	Open Sound System drivers.
metalab.unc.edu	Extensive Linux archive (formerly **sunsite.unc.edu**).
happypenguin.org	Linux Game Tome.
www.linuxgames.com	Linux games.
www.linuxquake.com	Quake.

Note Red Hat provides many RPM packaged applications as part of its Power Tools collection. Check the **powertools** directory for your release, or download the powertools disk image.

The software packages on RPM sites like Red Hat and **rpmfind.net** will have the file extension **.rpm**. RPM packages that contain source code have an extension **.src.rpm**. Other packages such as those in the form of source code that you will need to compile come in a variety of compressed archives. These will commonly have the extensions **.tar.gz** or **tar.bz2**. They are explained in detail later in the chapter. Table 31-2 lists several common file extensions that you will find for the great variety of Linux software packages available to you. See Chapter 32 for more details on archives and compression.

Table 31-2: Linux Software Package File Extensions

Extension	File
.rpm	Software package created with the Red Hat Software Package Manager (RPM).
.src.rpm	Software packages that are source code versions of applications, created with the Red Hat Software Package Manager (RPM).
.gz	gzip compressed file (use gnunzip to decompress, also **z** option with tar, as in **xvzf**).
.bz2	bzip2 compressed file (use bunzip2 to decompress, also **j** option with tar, as in **xvjf**).
.tar	A tar archive file, use tar with **xvf** to extract.
.tar.gz	gzip compressed tar archive file. Can use **z** option with **tar**, **tar xvzf** *archive*.
.tar.bz2	bzip2 compressed tar archive file. Can use **j** option with **tar**, **tar xvjf** *archive*.

Table 31-2: Linux Software Package File Extensions

Extension	File
tar.tz	tar archive file compressed with the **compress** command.
tar.Z	File compressed with the **compress** command (use the **decompress** command to decompress).
.deb	Debian Linux package.

Red Hat Package Manager (RPM)

Several Linux distributions, including Red Hat, OpenLinux, and SuSE, use RPM to organize Linux software into packages you can automatically install or remove. An RPM software package operates as its own installation program for a software application. A Linux software application often consists of several files that need to be installed in different directories. The program itself is most likely placed in a directory called **/usr/bin**, online manual files go in another directory, and library files in yet another directory. In addition, the installation may require modification of certain configuration files on your system. The RPM software package performs all these tasks for you. Also, if you later decide you don't want a specific application, you can uninstall packages to remove all the files and configuration information from your system. RPM works similarly to the Windows Install Wizard, automatically installing software, including configuration, documentation, image, sample, and program files, along with any other files an application may use. All are installed in their appropriate directories on your system. RPM maintains a database of installed software, keeping track of all the files installed. This enables you to use RPM also to uninstall software, automatically removing all files that are part of the application.

To install and uninstall RPM packages, you can use the **rpm** command on a shell command line or any available RPM window-based program, such as Kpackage or GnomeRPM. Although you should download RPM packages from your particular distribution, numerous RPM software packages are designed to run on any Linux system. Many of these are located at distribution **contrib** directories. You can learn more about RPM at its Web site at www.rpm.org. The site contains up-to-date versions for RPM, documentation, and RPM support programs, such as **rpm2html** and **rpm2cpio**. rpm2html takes a directory containing RPM packages and generates Web pages listing those packages as links that can be used to download them. **rpm2cpio** is Perl script to extract RPMs. You can obtain further documentation from the RPM Documentation Project site at www.rpmdp.org.

The RPM packages on your CD-ROMs only represent a small portion of the software packages available for Linux. You can download additional software in the form of RPM packages from distribution contribution locations, such as the **contrib** directory in the Red Hat FTP site at ftp.redhat.com. In addition, these packages are organized into **lib5** and **lib6** directories. **lib5** refers to the packages using the older libraries, whereas **lib6** refers to those using the current GNU 2.x libraries. For Red Hat 6.0 and later, you should use the **lib6** versions-though many packages still use the **lib5** versions, which also work.

An extensive repository for RPM packages is also located at http://rpmfind.net/linux/RPM. Packages here are indexed according to distribution, group, and name. It includes packages for every distribution, including Red Hat. From http://rpmfind.net, you can download the **rpmfind** command that enables you to search for RPM packages, either on your local system or on the RPM repository at **rpmfind.net**. You can even use **rpmfind** to download and update packages. **rpmfind** detects your system's distribution and lists RPM packages for it. Search results also tell you on what other packages a given RPM can depend. With the **--appropos** option, you can use more general terms to locate a package, instead of filename patterns. With the **--upgrade** option, you can download and install newer versions of installed packages. The **rpmfind** command also sets up a **.rpmfind** configuration file, where you can specify such features as a download directory, the remote servers to search, and the location of local RPM packages on your system.

Note RPM packages with the term **noarch** are used for architecture-independent packages. This means that they are designed to install on any Linux distribution. Packages without **noarch** may be distribution dependent, with some designed to install on Red Hat and others for distributions like Caldera and Debian.

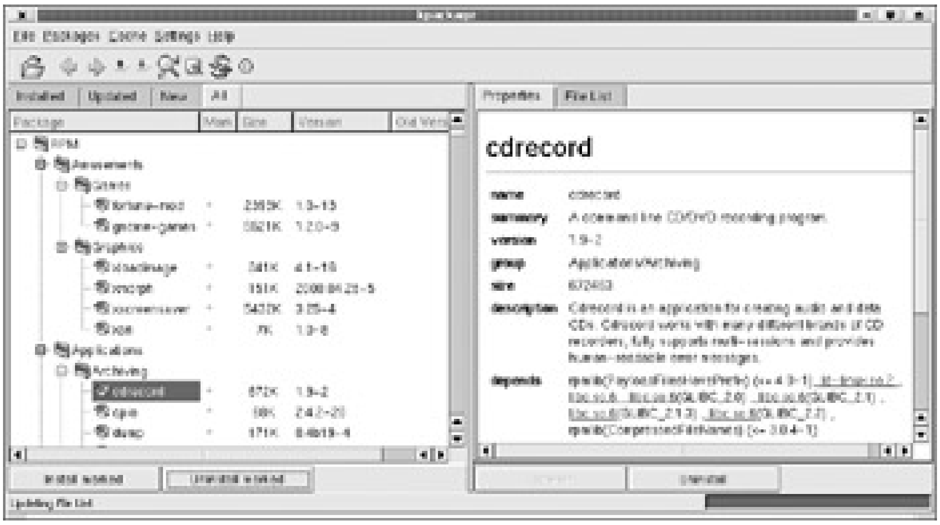
Your Red Hat distribution CD-ROM included with this book contains an extensive set of applications located in an **RPMS** directory on the CD-ROM, **RedHat/RPMS**. You can install or uninstall any of these packages using an **rpm** command, a GUI RPM utility, or the GMC Gnome desktop. To install a software package from your CD-ROM using the **rpm** command, it is easier to move first to the **RPMS** directory and then install the package you want. Be sure to mount the CD-ROM first before you try to access it.

You can download additional RPM packages not located on your CD-ROM from the Red Hat FTP site at **ftp.redhat.com**. Web sites for the particular software you want may also have RPM packages already set up for you for Red Hat. For example, you can obtain the ProFTPD RPM package for Red Hat from **ftp.redhat.com**, and the current Red Hat or Linuxconf RPM packages from the Linuxconf Web site (see Chapter 30). You could place these packages in a directory on your system, and then use either **rpm** or a GUI RPM utility such as GnomeRPM to install it. Normally, you should always try to use the version of the RPM package set up for your distribution. In many cases, attempting to install an RPM package meant for a different distribution may fail. Popular RPM package managers are listed here:

Kpackage	K Desktop RPM package manager
GnomeRPM	GnomeRPM package manager
rpm	The shell command to manage RPM packages

The K Desktop Package Manager: kpackage

The KDE desktop provides a powerful and easy-to-use RPM package manager called *Kpackage* (see Figure 31-1). You run Kpackage under any window manager or desktop (including Gnome), as long as you have installed the K Desktop on your system. You can start Kpackage by selecting its entry in the K menu Utility menu or by entering the **kpackage** command in a terminal window.



cdrecord

name cdrecord
summary A command line CD/DVD recording program.
version 1.9-2
group Applications/Archiving
size 672453
description Cdrecord is an application for creating audio and data CDs. Cdrecord works with many different brands of CD recorders, fully supports multi-sessions and provides human-readable error messages.

Figure 31-1: The KDE Kpackage RPM package manager

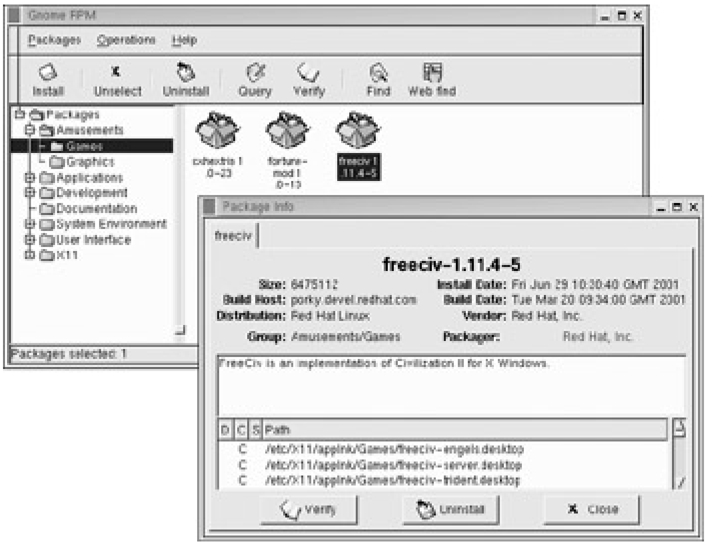
The right side of Kpackage contains two tabbed panels: one for Properties and the other for the File List. The Properties panel displays information about the software in the currently selected RPM package, including the version number and the authors. The File List panel lists all the files contained in the software package, including **README** files. If you are using Kpackage on the K Desktop, you can click any text file in the File List, and it is displayed by the text editor. This is a convenient way to read installation files, such as **README** or **install** files. To uninstall a package, select it and click Uninstall.

The list of installed packages is often extensive. To locate a particular package, select the Find entry in the File menu. If the Substring check box is selected, you can use a pattern to search for your package instead of a complete name. The Package list in the left pane moves to the first package found, highlighting it. If you use a pattern and more choices exist, you can move to the next one by selecting the Find menu item again. The Kpackage application also enables you to search for a particular file in a package. Select the Find File entry in the File menu and enter the name of the file to locate. You have to use the full pathname for the file. Patterns are not supported.

You can also use Kpackage to install RPM packages. You must know where on the system the packages are located. For example, on Red Hat systems, they are in the **RedHat/RPMS** directory. If you mounted the CD-ROM at **/mnt/cdrom**, the full pathname should be **/mnt/cdrom/RedHat/RPMS**. Packages you download from FTP sites would be in whatever directory you downloaded them to-say, **/root/download**. To install a package, select Open from the File menu. This opens a File Browser dialog box where you can move to the directory you want and select the RPM package. An Installation dialog box is displayed with options that include updating and testing the package. You select Update for packages that are updated versions of ones already installed. With the **Test** option, you can test an installation without actually having to install it.

GnomeRPM

Although not written by Red Hat, GnomeRPM provides an effective and easy-to-use interface for managing RPM packages on your Gnome desktop. It runs on any window manager, provided Gnome is installed on your system. As Figure 31-2 shows, the GnomeRPM window displays two panes, the left one showing a tree listing categories of different installed RPM packages. Expand a category to display the packages in the right pane. You can query a package by selecting it and clicking the Uninstall icon in the icon bar, or by right-clicking it and selecting Uninstall from the pop-up menu. You can use the same method for querying packages and for displaying information and file listings. You can select several packages at once from different categories by CTRL-clicking their icons. A selected package darkens. When you select uninstall, all those packages are uninstalled. CTRL-click the package again to deselect it. Click Unselect to deselect all the packages you selected. The number of selected packages is shown in the lower-left corner of the window. The GnomeRPM package also features a find utility, which you can use to locate RPM packages easily. In the find window, you can then query or uninstall the package.



Packages selected: 1

freeciv-1.11.4-5

Size: 6475112
Install Date: Fri Jun 29 10:30:40 GMT 2001
Build Host: porky.devel.redhat.com
Build Date: Tue Mar 20 09:34:00 GMT 2001
Distribution: Red Hat Linux
Vendor: Red Hat, Inc.
Group: Amusements/Games
Packager: Red Hat, Inc.

FreeCiv is an implementation of Civilization II for X Windows.

D	C	S	Path
	C		/etc/X11/applnk/Games/freeciv-engels.desktop
	C		/etc/X11/applnk/Games/freeciv-server.desktop
	C		/etc/X11/applnk/Games/freeciv-trident.desktop

Figure 31-2: The GnomeRPM main window

To install new packages with GnomeRPM, click the Install icon. This opens a window that displays the selected packages to install. You then click Add to open a window for locating packages on your system. You can add as many windows as you want to the list. Click Install to install the packages. To upgrade a package that is already installed, click Upgrade and follow the same procedure.

GnomeRPM supports drag-and-drop operations from the Gnome desktop and file manager. You can drag one or more RPM packages from a file manager window to the GnomeRPM install dialog window. You can then install or query selected packages. GnomeRPM also enables you to browse through packages available with the **rpmfind.net** system. Click the Web Find button. The listing of packages is then downloaded and displayed in the tree menu. Use the tree menu to navigate to the package you want.

KDE and Gnome File Managers

On Gnome, you can install RPM packages directly from the GMC file manager, without starting up a special utility such as GnomeRPM (Nautilus does not yet support this feature). Use the GMC file manager window to access the directory with your package, such as your distribution CD-ROM. Then right-click the package name or icon. In the pop-up menu, you can select the Install option to install the package, or Update to update it. You can also simply just double-click the package. The Gnome file manager automatically detects that it is an RPM package, checks to see if it is installed, and then installs it for you. It will also perform a dependency check to see if there are any conflicts or needed libraries. If there are any problems, it will halt the installation and display any conflict or dependency warnings. You can use this same method for FTP sites. The Gnome file manager is Internet-aware. You can enter a URL for an FTP site in its Location box to access the site. Be sure to include the FTP protocol in the URL, **ftp://**. When you locate the package, right-click it and select Install or Update. The file is downloaded and then automatically installed on your system.

Note On Gnome, if you just want to find out what a package is, along with a list of files in that package, you can right-click on the package and select the Show Info entry. A window opens up displaying a description of the software and a list of all the files in the package. There are also buttons to let you install or upgrade the software.

Problems can arise either if the application you want to install is already installed or if it requires that other application be installed first (upgrades are considered new software and are not considered a problem). In this case a window titled "Installation Problems" will appear asking you "Do you want to ignore these problems". The window will display a list of problems encountered, such as the software is already installed or it requires other software packages to be installed first. You should cancel the install operation by clicking on the NO button, or, if the NO button is selected, press ENTER. You do *not* want to ignore the problems. Clicking YES instead will force the installation of the software anyway, which may overwrite previous configuration files or let you try to run the application without required supporting software such as needed libraries.

The KDE file manager also enables you to install RPM packages, though it uses Kpackage to perform the actual installation. Locate an RPM file on your system using the file manager, and then single-click its icon or name. This automatically opens Kpackage with the RPM package loaded, which is then displayed on a window with panels for the package information and the list of files in it. At the bottom of the panels is an Install button. Click it to install the package. Because the KDE file manager is Internet-aware, you can use this same method both to download and install RPM packages from FTP sites. Locate the FTP site with the file manager (enter its URL in the Location box), and then locate your package. Once your package is listed in the file manager window, click it. The package is then automatically downloaded and Kpackage starts up, showing the package. Click the Install button to install it.

Updating Software

As noted in Chapter 4, you can update your Red Hat system automatically using the Red Hat Update Agent. You can also download packages using an FTP client, Web browser, or the Gnome or KDE file managers, and then use the **rpm** command to install the software. Also, you can use Kpackage or Gnome RPM to access the distribution FTP sites directly, download the package, and automatically install it on your system. Just enter the FTP URL for the site in the Location box.

For the **rpm** command, you use the **-U** option to upgrade packages. In the following example, the **rpm** command with the **-Uvh** option installs an upgrade for Linuxconf:

```
$ rpm -Uvh  linuxconf-1.25r7-1.i386.rpm
```

Webmin and Linuxconf

With either Webmin or Linuxconf, you can also manage software installations. For Webmin, select the Software Packages icon in the System page. From the Software Packages page, you can then view installed packages or install new ones, as well as search for packages. Clicking an entry for an installed package will display a page showing package information as well as a button for uninstalling the package. You can install new packages from your local system or from remote FTP sites.

The recent version of Linuxconf also provides software management modules. On the Control tab open the Package Management (RPM) entry to list several installation tasks. You can install just one package or several packages in a specified directory. You can also browse uninstalled packages, selecting the one you want. You will have to specify the directory in which they are located. Use the Browse Installed Packages entry to list all your installed packages by category. Here you can obtain information about a package, as well as uninstall it.

Command Line Installation: rpm

If you do not have access to the desktop or you prefer to work from the command line interface, you can use the **rpm** command to manage and install software packages. **rpm** is the command that actually performs installation, removal, and queries of software packages. In fact, GnomeRPM and Kpackage use the **rpm** command to install and remove packages. An RPM package is an archive of software files that include information about how to install those files. The filenames for RPM packages end with **.rpm**, indicating software packages that can be installed by the Red Hat Package Manager.

With the **rpm** command, you can maintain packages, query them, build your own, and verify the ones you have. Maintaining packages involves installing new ones, upgrading to new versions, and uninstalling packages. The **rpm** command uses a set of options to determine what action to take. In addition, certain tasks, such as installing or querying packages, have their own options that further qualify the kind of action they take. For example, the **-q** option queries a package, but when combined with the **-l** option it lists all the files in that package. Table 31-3 lists the set of **rpm** options.

Table 31-3: Red Hat Package Manager (RPM) Options

Mode of Operation	Effect
rpm -i*options package-file*	Installs a package; the complete name of the package file is required.
rpm -e*options package-name*	Uninstalls (erases) a package; you only need the name of the package, often one word.
rpm -q*options package-name*	Queries a package; an option can be a package name or a further option and package name, or an option applied to all packages.
rpm -U*options package-name*	Upgrade; same as install, but any previous version is removed.
rpm -bO*options package-specifications*	Builds your own RPM package.

Table 31-3: Red Hat Package Manager (RPM) Options

Mode of Operation	**Effect**
rpm -F*options package-name*	Upgrade, but only if package is currently installed.
rpm -verify*options*	Verifies a package is correctly installed; uses same options as query; you can use **-V** or **-y** in place of **-verify**.
--nodeps	Installs without doing any dependency checks.
--force	Forces installation despite conflicts.
--percent	Displays percentage of package during installation.
--test	Tests installation; does not install, only checks for conflicts.
-h	Displays # symbols as package is installed.
--excludedocs	Excludes documentation files.
Uninstall Option (to be used with -e)	
--test	Tests uninstall; does not remove, only checks for what is to be removed.
--nodeps	Uninstalls without checking for dependencies.
--allmatches	Removes all versions of package.
Query Option (to be used with -q)	
package-name	Queries package.
-qa	Queries all packages.
-qf *filename*	Queries package that owns *filename*.
-qR	List packages on which this package depends.
-qp *package-name*	Queries an uninstalled package.
-qi	Displays all package information.
-ql	Lists files in package.
-qd	Lists only documentation files in package.
Query Options (to be used with -q)	
-qc	Lists only configuration files in package.
-q --dump	Lists only files with complete details.
General Options (to be used with any option)	
-vv	Debug; displays descriptions of all actions taken.
--quit	Displays only error messages.
--version	Displays **rpm** version number.
--help	Displays detailed use message.
--root*directory*	Uses directory as top-level directory for all operations (instead of root).
--dbpath*directory*	Uses **RPM** database in the specified directory.
--dbpath *cmd*	Pipes output of RPM to the command **cmd**.
--rebuilddb	Rebuilds the **RPM** database; can use with **-root** and **-dbpath** options.

Table 31-3: Red Hat Package Manager (RPM) Options

Mode of Operation	**Effect**
--initdb	Builds a new **RPM** database; **-root** and **-dbpath** options.
Other Sources of Information	
RPM-HOWTO	More detailed information, particularly on how to build your own RPM packages.
man rpm	Detailed list of options.
www.rpm.org	RPM Web site with latest RPM software.
www.rpmdp.org	RPM Documentation Project.

You use the **-i** option to install a new software package and the **-U** option to update a currently installed package with a newer version. With a **-e** option, **rpm** uninstalls the package. The -**q** option tells you if a package is already installed, and the **-qa** option displays a list of all installed packages. Piping this output to a pager utility, such as more, is best.

```
rpm -qa | more
```

In the next example, the user checks to see if mozilla is already installed on the system. Notice the full filename of the RPM archive is unnecessary. If the package is installed, your system has already registered its name and where it is located.

```
# rpm -q mozilla
Mozilla-0.7015
```

Note Keep in mind the distinction between the installed software package name and the package filename. The filename will end in a **.rpm** extension and can only be queried with a **p** option.

You can combine the **q** options with the **i** or **l** option to display information about the package. The options **-qi** display information about the software, such as the version number or author (**-qpi** queries an uninstalled package file). The option **-ql** displays a listing of all the files in the software package. The **--h** option provides a complete list of **rpm** options. Common query options are shown here.

-q application	Checks to see if an application is installed.
-qa *application*	Lists all installed RPM applications.
-qf *filename*	Queries application that owns *filename*.
-qR *application*	List applications on which this application depends.
-qi *application*	Displays all application information.
-ql *application*	Lists files in application.
-qd *application*	Lists only documentation files in application.
-qc *application*	Lists only configuration files in application.

If you want to query an RPM package file, a file ending with **.rpm**, you use the same query options, but with the **p** option added, as shown here:

-qpi RPM-file	Displays all package information in the RPM package.
-qpl *RPM-file*	Lists files in the RPM package.
-qpd *RPM-file*	Lists only documentation files in the RPM package.
-qpc *RPM-file*	Lists only configuration files in the RPM package.
-qpR *RPM-file*	List packages on which this RPM package depends.

If your RPM query outputs a long list of data, like an extensive list of files, you can pipe the output to the **more** command to look at it screen by screen, or even redirect the output to a file.

```
rpm -ql mozilla | more
rpm -qpl openmotif-2.1.30-1_ICS.386.rpm  > mytemp
```

The syntax for the **rpm** command is as follows (*rpm-package-name* is the name of the software package you want to install):

```
rpm options rpm-package-name
```

Note The software package filename is usually lengthy, including information about version and release in its name. All end with **.rpm**.

If you are installing from a CD-ROM, you can change to the CD-ROM's **RedHat/RPMS** directory, which holds the RPM packages. An **ls** command lists all the software packages. If you know how the name of a package begins, you should include that with the **ls** command and an attached *. The list of packages is extensive and does not all fit on one screen. This is helpful for displaying the detailed name of the package. The following example lists most X Window System packages:

```
# ls x*
```

You use the **-i** option to install new packages and the **-U** option to update currently installed packages with new versions. If you try to use the **-i** option to install a newer version of an installed package, you receive an error saying the package is already installed. In the next example, the user first installs a new package with the **-i** option, and then updates a package with the **-U** option. Including the **-v** and **-h** options is customary. Here, **-v** is the verbose option that displays all files as they are installed, and **-h** displays a cross-hatch symbol periodically to show RPM is still working.

In the following example, the user installs the software package for the XV screen capture program available from Red Hat **powertools** directory or from **rpmfind.net**. Notice the full filename is entered. To list the full name, you can use the **ls** command with the first few characters and an asterisk, **ls** htdig*. The **h** option displays # symbols as the installation takes place. The **rpm** command with the **-q** option is then used to check that the software was installed. For installed packages only, the software name needs to be used-in this case, xv-3.10a-23.

```
 [root@turtle mypackages]# ls xv*
xv-3.10a-23.i386.rpm
[root@turtle mypackages]# rpm -ivh xv-3.10a-23.i386.rpm
xv-3.10a-23                  ########################################
[root@turtle mypackages]# rpm -q xv
```

```
xv-3.10a-23
```

To display information about the installed package, use **-qi**, and **-ql** displays a listing of the files a given RPM package contains.

```
# rpm -qi xv
# rpm -ql xv
```

To display information taken directly from an RPM package, you add the **p** qualifier to the **q** options. The **-qpi** combination displays information about a specific package, and **-qpl** displays a listing of the files a given RPM package contains. In this case, you must specify the entire filename of the RPM package. You can avoid having to enter the entire name simply by entering a unique part of the name and using the * filename matching character to generate the rest.

```
[root@turtle mypackages]# ls proftp*
proftpd-core-1.2.0pre10-1.i386.rpm
[root@turtle mypackages]# rpm -qp proftpd-core-1.2.0pre10-1.i386.rpm
proftpd-1.2.0pre3-2
[root@turtle mypackages]# rpm -qpi proftpd-core-1.2*.rpm
Name        : proftpd                     Relocations: (not relocateable)
Version     : 1.2.0pre10                        Vendor: (none)
........................................................
[root@turtle mypackages]# rpm -qpl proftpd*
/etc/logrotate.d/proftpd
/etc/pam.d/ftp
/etc/proftpd.conf
....................................................
```

Remember, if you are installing an upgrade, you need to use the **-U** option instead of the **-i** option. If you try to use **-i** to upgrade a package, you receive an error saying the package is already installed.

```
# rpm -Uvh mozilla-0.7-15.i386.rpm
```

If you receive an error stating dependency conflicts exist, the package may require other packages or their updated versions to be installed first. In some cases, you may have to install with the no dependency check options, **--nodeps** (notice the two dashes before the option). In some rare cases, installation instructions for a particular package may require you to use **--nodeps**. Another risky option is the **--force** option. This forces installation, overwriting any current files. This is a brute-force approach that should be used with care, only as a last resort.

If you are worried that a software package will install on your system incorrectly, you can use the test option(**--test**) in the debut mode (**vv**) to see exactly what actions RPM will take.

```
# rpm -ivv --test xv-3.10a-23.i386.rpm
```

Note A few RPM packages, like those for OpenOffice, are designed only to extract a subdirectory of install binaries with its own install program. For example, OpenOffice uses its own installation program called setup in its subdirectory. Use **./setup** to run it.

To remove a software package from your system, first use **rpm -q** to make sure it is actually installed. Then, use the **-e** option to uninstall it. You needn't use the full name of the installed file. You only need the name of the application. For example, if you decide you do not need XV, you can remove it using the **-e** option and the software name, as shown here:

```
# rpm  -e  xv
```

If direct conflicts occur with another software package, you may have to uninstall the other package first. This is the case with wu-ftpd and ProFTP on many distributions. Man distributions currently install wu-ftpd as the default FTP server. You must first uninstall the wu-ftpd with the **-e** option before you can install ProFTP. However, when you try to do this, you receive a dependency error. You can overcome this error by using the **--nodeps** option. Once wu-ftpd is removed, you can install ProFTP.

```
[root@turtle mypackages]# rpm -e --nodeps wu-ftpd
[root@turtle mypackages]# rpm -ivh proftpd-core-1.2*rpm
proftpd       #############################################
[root@turtle mypackages]# rpm -q proftpd
proftpd-core-1.2.0pre3-10
```

You can use the verify option (**-V**) to check to see if any problems occurred with the installation. RPM compares the current attributes of installed files with information about them placed in the RPM database when the package was installed. If no discrepancies exist, RPM outputs nothing. Otherwise, RPM outputs a sequence of eight characters, one for each attribute, for each file in the package that fails. Those that do not differ have a period. Those that do differ have a corresponding character code, as shown here:

5	MD5 checksum
S	File size
L	Symbolic link
T	File modification time
D	Device
U	User
G	Group
M	Mode (includes permissions and file types)

The following example verifies the proftpd package:

```
[root@turtle mypackages]# rpm -V proftpd
```

To compare the installed files directly with the files in an RPM package file, you use the **-Vp** option, much like the **-qp** option. To check all packages, use the **-Va** option as shown here:

```
rpm -Va
```

If you want to verify a package, but only know the name of a file in it, you can combine verify with the **-f** option. The following example verifies the RPM package containing the **ftp** command:

```
rpm -Vf  /bin/ftp
```

A complete description of **rpm** and its capabilities is provided in the online manual.

```
# man rpm
```

RPM maintains a record of the packages it has installed in its **RPM** database. You may, at times, have to rebuild this database to ensure RPM has current information on what is installed and what is not. Use the **--rebuilddb** options to rebuild your database file.

```
rpm --rebuilddb
```

To create a new RPM database, use the **--initdb** option. This can be combined with **--dbpath** to specify a location for the new database.

Installing Software from Red Hat RPM Source Code Files: SRPMS

Red Hat and several other distributors also make available source code versions of their binary RPM packaged software. The source code is packaged into RPM packages that will be automatically installed into designated directories where you can easily compile and install the software. Such packages are called SRPMs and have the added suffix **src**. Such files end in the suffix **.src.rpm**. Source code versions for packages in the Red Hat distribution are located on Red Hat releases in the SRPMS directory. Many online sites like **rpmfind.net** will also list SRPM packages. Source code versions have the advantage of letting you to make your own modifications to the source code, allowing you to generate your own customized versions of RPM packaged software. You still use the **rpm** command with the **-i** option to install source code packages. In the following example, the user installs the source code for xpuzzles:

```
rpm -i xpuzzles-5.5.2-4.src.rpm
```

Red Hat SRPM files are installed in various subdirectories in the **/usr/src/redhat** directory. When Red Hat SRPMs are installed, a spec file placed in the **/usr/src/redhat/ SPECS** directory and the compressed archive of the source code files is placed in the **/usr/src/redhat/SOURCES** directory. For xpuzzles, a spec file called **xpuzzles.spec** was placed in **/usr/src/redhat/SPECS**, and a compressed archive called **xpuzzles-5.5.2.tar.gz** was placed in the **/usr/src/rehat/SOURCES** directory (spec files are discussed in more detail in the following section on building RPM packages).

You now need to build the source code files, extracting them and running any patches on them that may be included with the package. You do this with a single **rpm** command run on the SPEC file using the -bp option. Change to the **/usr/src/redhat/SPECS** directory and use the **rpm** command again, this time with the **-bp** option, to generate the source code files.

```
cd /usr/src/redhat/SPECS
rpm -bp xpuzzles.spec
```

The resulting source code files are placed in their own subdirectory with the package's name in the **/usr/src/redhat/BUILD** directory. For xpuzzles, the xpuzzle source code is placed in **/usr/src/redhat/BUILD/xpuzzles-5.5.2** directory. In this subdirectory you can then modify the source code, as well as compile and install the application. Check the **README** and **INSTALL** files for details.

Installing Software from Compressed Archives: .tar.gz

Linux software applications in the form of source code are available at different sites on the Internet. You can download any of this software and install it on your system. You download software using an FTP client as described in Chapter 12. Recent and older software is usually downloaded in the form of a compressed archive file. Applications will always be downloadable as compressed archives, should they not have an RPM version. This is particularly true for the recent versions of Gnome or KDE packages. RPM packages are only intermittently generated. A *compressed archive* is an archive file created with tar, and then compressed with gzip. To install such a file, you must first decompress it with the gunzip utility, and then use tar to extract the files and directories making up the software package. Instead of the gunzip utility, you could also use **gzip -d**. The next example decompresses the **htdig-3.1.5.tar.gz** file, replacing it with a decompressed version called **htdig-3.1.5.tar**:

```
$ ls
 htdig-3.1.5.tar.gz
$ gunzip htdig-3.1.5.tar.gz
$ ls
htdig-3.1.5.tar.gz
```

First, use **tar** with the **t** option to check the contents of the archive. If the first entry is a directory, then that directory is created and the extracted files are placed in it. If the first entry is not a directory, you should first create one and then copy the archive file to it. Then extract the archive within that directory. If no directory exists as the first entry, files are extracted to the current directory. You must create a directory yourself to hold these files.

```
$ tar tvf htdig-3.1.5.tar
```

Now you are ready to extract the files from the tar archive. You use **tar** with the **x** option to extract files, the **v** option to display the pathnames of files as they are extracted, and the **f** option, followed by the name of the archive file:

```
$ tar xvf htdig-3.1.5.tar
```

You can combine the decompressing and unpacking operation into one **tar** command by adding a **z** option to the option list, **xzvf**. The following command both decompresses and unpacks the archive:

```
$ tar xzvf htdig-3.1.5.tar.gz
```

The extraction process will create a subdirectory consisting of the name and release of the software. In the previous example, the extraction created a subdirectory called **htdig-3.1.5**. You can then change to that directory to access the software files.

```
$ cd htdig-3.1.5
```

Installation of your software may differ for each package. Instructions are usually provided along with an installation program. See the following section on compiling software for information on how to create and install the application on your system.

Downloading Compressed Archives from Online Sites

Many software packages under development or designed for cross-platform implementation may not be in an RPM format. Instead, they may be archived and compressed (see Chapter 32). The filenames for these files end with the extensions **.tar.gz**, **.tar.bz2**, or **.tar.Z**. The different extensions indicate different decompression methods using different commands: **gunzip** for **gz**, **bunzip2** for **bz2**, and **decompress** for **Z**. In fact, most software with an RPM format also has a corresponding **.tar.gz** format. After you download such a package, you must first decompress it, and then unpack it with the **tar** command. For the **.gz** files, you use **gunzip**, and for **.bz2** files you can use **bunzip2**. The compressed archives could hold either source code that you then need to compile, or, as is the case with Java packages, binaries that are ready to run.

You can download compressed archives from many different sites, including those mentioned previously. Downloads can be accomplished with FTP clients like ncftp and Gftp, or with any Web browser like mozilla. Once downloaded, any file that ends with a **.Z** , **bz2**, **.zip**, or **.gz** is a compressed file that must be decompressed. In the following example, the **gunzip** command is used to decompress the CD-Rchive CD writer downloaded from **apps.kde.com**.

```
# gunzip cdrchive-1.2.2.tar.gz
```

For files ending with **bz2** you would use the **bunzip2** command. The following example decompresses the Java 2 SDK downloaded through **www.blackdown.org**:

```
# bunzip2 j2sdk-1.3.0-FCS-linux-i386.tar.bz2
```

If the file then ends with **.tar**, it is an archived file that must be unpacked using the **tar** command. Before you unpack the archive, move it to the directory where you want it. Source code you intend to compile is usually placed in the **/usr/local/src** directory. Packages that hold binary programs ready to run, like Java, are meant to be extracted in certain directories. Usually this is the **/usr/local** directory. Most archives, when they unpack, create a subdirectory they named with the application name and its release, placing all those files or directories making up the software package into that subdirectory. For example, the file **cdrchive-1.2.2.tar** unpacks to a subdirectory called **cdrchive-1.2.2**. In certain cases, the software package that contain precompiled binaries is designed to unpack directly into the system subdirectory where it will be used. For example, it is recommended that **j2sdk-1.3.0-FCS-linux-i386.tar** be unpacked in the **/usr/local** directory where it will create a subdirectory called **j2sdk-1.3.0**. The **/usr/local/j2sdk-1.3.0/bin** directory will hold the Java binary programs. To check if an archive unpacks to a directory, use **tar** with the **t** option to list its contents and to see if the names are prefixed by a directory. If so, that directory is created and the extracted files are placed in it. If no directory name exists, create one and then copy the archive file to it. Then extract the archive within that directory.

```
# tar tf j2sdk-1.3.0-FCS-linux-i386.tar
```

Now you are ready to extract the files from the tar archive (see Chapter 32). You use **tar** with the **x** option to extract files, the **v** option to display the pathnames of files as they are extracted, and the **f** option, followed by the name of the archive file:

```
# tar xvf j2sdk-1.3.0-FCS-linux-i386.tar
```

This will create a subdirectory called j2sdk-1.3.0. You can change to this subdirectory and examine its files, such as the **README** and **INSTALL** files.

```
# cd j2sdk-1.3.0
```

The tar utility provides decompression options you can use to have **tar** first decompress a file for you, invoking the specified decompression utility. The **z** options will automatically invoke gunzip to unpack a **.gz** file, and the **j** option will unpack a **.bz2** file. Use the **Z** options for **.Z** files. The next example shows how you can combine decompression and extraction in one step:

```
# tar xvjf j2sdk-1.3.0-FCS-linux-i386.tar.bz2
# tar xvzf cdrchive-1.2.2.tar.gz
```

Installation of your software may differ for each package. Instructions are usually provided, along with an installation program. Downloaded software usually includes **README** files or other documentation. Be sure to consult them.

Compiling Software

Some software may be in the form of source code that you need to compile before you can install it. This is particularly true of programs designed for cross-platform implementations. Programs designed to run on various Unix systems, such as Sun, as well as on Linux, may be distributed as source code that is downloaded and compiled in those different systems. Compiling such software has been greatly simplified in recent years by the use of configuration scripts that automatically detect a given system's configuration and compile the program accordingly. For example, the name of the C compiler on a system could be **gcc** or **cc**. Configurations scripts detect which is present and use it to compile the program.

First, change to the directory where the software's source code has been extracted to.

```
# cd /usr/local/src/cdrchive-1.2.2
```

Before you compile software, read the **README** or **INSTALL** files included with it. These give you detailed instructions on how to compile and install this particular program. If the software used configuration scripts, then compiling and installing usually involves only the following three simple commands:

```
# ./configure
#  make
#  make install
```

Note Be sure to remember to place the period and slash before the **configure** command. **./** references a command in the current working directory, rather than another Linux command.

The **./configure** command performs configuration detection. The **make** command performs the actual compiling, using a **Makefile** script generated by the **./configure** operation. The **make install** command installs the program on your system, placing the executable program in a directory, such as **/usr/local/bin,** and any configuration files in **/etc**. Any shared libraries it created may go into **/usr/local/lib**.

Certain software may have specific options set up for the **./configure** operation. To find out what these are, you use the **./configure** command with the **--help** option.

```
./configure --help
```

A useful common option is the **-prefix** option, which lets you specify the install directory.

```
./configure -prefix=/usr/bin
```

If you are compiling an X-, Gnome-, or KDE-based program, be sure their development libraries have been installed. For X applications, be sure the xmkmf program is also installed. If you chose a standard install when you installed your distribution system, these most likely were not installed. For distributions using RPM packages, these come in the form of a set of development RPM packages, usually having the word "development" or "develop" in their name. You need to install them using either RPM, kpackage, or GnomeRPM. Gnome, in particular, has an extensive set of RPM packages for development libraries. Many X applications may need special shared libraries. For example, some applications may need the xforms library or the qt library. Some of these you need to obtain from online sites.

Some older X applications use xmkmf directly instead of a configure script to generate the needed **Makefile**. In this case, enter the command **xmkmf** in place of **./configure**. Be sure to consult the **INSTALL** and **README** files for the software. Usually, you only need to issue the following commands within the directory that contains the source code files for the software:

```
xmkmf
make
make install
```

If no configure script exists and the program does not use **xmkmf,** you may have to enter the **make** command, followed by a **make install** operation. Check the **README** or **INSTALL** files for details.

```
make
make install
```

Be sure to check the documentation for such software to see if any changes must be made to the **Makefile**. Only a few changes may be necessary, but more detailed changes require an understanding of C programming and how **make** works with it. If you successfully configure the **Makefile**, you may only have to enter the **make** and **make install** operations. One possible problem is locating the development libraries for C and X Windows. X Windows libraries are in the **/usr/X11R6/lib** directory. Standard C libraries are located in the **/usr/lib** directory. You can set the PATH variable to hold the pathnames for any special libraries as described in the next section.

Once you have compiled and installed your application, and have checked that it is working properly, you can remove the source code directory that was created when you extracted the software. You can keep the archive file (tar) in case you need to extract the software again. Use **rm** with the **-rf** options so all subdirectories will be deleted and you do not have to confirm each deletion.

```
rm -rf cdrchive.1.2.2
```

Command and Program Directories: PATH

Programs and commands are usually installed in several standard system directories, such as **/bin**, **/usr/bin**, **/usr/X11R6/bin**, or **/usr/local/bin**. Some packages place their commands in subdirectories, however, which they create within one of these standard directories or in an entirely separate directory. In such cases, you may be unable to run those commands because your system may be unable to locate them in the new subdirectory. Your system maintains a set of directories that search for commands each time you execute one. This set of directories is kept in a system variable called PATH that is created when you start your system. If a command is in a directory that is not in this list, your system will be unable to locate and run it. To use such commands, you first need to add the new directory to the set of directories in the PATH variable.

Note On Red Hat systems, the PATH variable is originally assigned in the **/etc/rc.d/rc.sysinit** file, and further added to by different services that start up when the system boots. You could edit **/etc/rc.d/rc.sysinit** file directly, but you would have to be very careful not to change anything else. A safer approach is to add a **PATH** definition in the **/etc/profile** file.

To add a directory, you would add a PATH entry to the **/etc/profile** file, which is run whenever users log in to their accounts. Carefully edit the **/etc/profile** file using a text editor, such as kedit, gedit, Emacs, or Vi (you may want to make a backup copy first with the **cp** command). You add a line that begins with **PATH**, followed by an = sign, and the term **$PATH**, followed by a colon, and then the directory to be added. The **$** before **PATH** is critically important. If you add more than one directory, be sure a colon separates them. You should also have a colon at the end. For example, if you install the Java 2 SDK, the Java commands are installed in a subdirectory called **j2sdk-1.3.0/bin** in the **/usr/local** directory. The full pathname for this directory is **/usr/local/j2sdk-1.3.0/bin**. You need to add this directory to the list of directories assigned to PATH in the **/etc/profile** file. The following example shows the PATH variable with its list of directories and the **/usr/local/j2sdk-1.3.0/bin** directory added. Notice the **$** before **PATH** after the = sign, **PATH=$PATH**.

```
PATH=$PATH:/usr/local/j2sdk-1.3.0/bin
```

The **/etc/profile** script is a system script executed for each user when the user logs in. Individual users can customize their PATH variables by placing a PATH assignment in either their **.bashrc** or **.bash_profile** files. In this way, users can access commands and programs they create or install for their own use in their own user directories (see Chapter 13 for more details). On Red Hat, user **.bash_profile** files will already contain the following PATH definition. Notice the use of $PATH, which keeps all the directories already added to the PATH in previous startup scripts like **/etc/profile** and **/etc/rc.d/ rc.sysinit**.

```
PATH=$PATH:$HOME/bin
```

The following entry in the **. bash_profile** file adds a user's **newbin** directory to the PATH variable. Notice both the colon placed before the new directory and the use of the **$HOME** variable to specify the pathname for the user's home directory.

```
PATH=$PATH:$HOME/bin/:$HOME/newbin
```

In the **.bash_profile** file for the **root** user, the PATH definition also includes **sbin** directories. The **sbin** directories hold system administration programs that the root user would need to have access to. The **root** user **PATH** is shown here:

```
PATH=/usr/local/sbin:/usr/sbin:/sbin:$PATH:$HOME/bin
```

Packaging Your Software: Building RPMs

Once you finish developing your software, you may then want to distribute it to others. Ordinarily, you would pack your program into a tar archive file. People would then download the file and unpack it. You would have to include detailed instructions on how to install it and where to place any supporting documentation and libraries. Any number of variations might stop installation of a program.

The RPM is designed to automate these tasks. RPM automatically installs software on a system in the designated directories, along with any documentation, libraries, or support programs. It has complex and powerful capabilities, and can handle the most complex programs. Simple examples of its use are provided here.

Note The Autoconf program is used to configure the source code for an application automatically to a given system.

Creating RPM Packages

The package creation process is designed to take the program through several stages, starting with unpacking it from an archive, and then compiling its source code, and, finally, generating the RPM package. You can skip any of these stages, up to the last one. If your software is already unpacked, you can start with compiling it. If your software is compiled, you can start with installation. If it is already installed, you can go directly to creating the RPM package.

RPM makes use of three components to build packages: the build tree, the **rpmrc** configuration files, and an **rpm** spec script. The build tree is a set of special instructions used to carry out the different stages of the packaging process. The **rpm** spec script contains instructions for creating the package, as well as the list of files to be placed in it. The **rpmrc** files are used to set configuration features for RPM. The **/usr/lib/rpm/ rpmrc** file holds the default options for your system and is always read. You can also set up a **/etc/rpmrc** file for global options you want to set for your system. Entries here will override those in the **/usr/lib/rpm/rpmrc** file. You can also set up a local **.rpmrc** file in your home directory, which will override both of these. To obtain a listing of the **/usr/lib/rpm/rpmrc** file, enter

```
$ rpm --showrc
```

The build tree directories, listed in the following table, are used to hold the different files generated at each stage of the packaging process. The **SOURCES** directory holds the compressed archive. The **BUILD** directory holds the source code unpacked from that archive. The **RPMS** directory is where the RPM package containing the executable binary program is placed, and **SRPMS** is where the RPM package containing the source code is placed. If you are creating a package from software stored in a compressed archive, such as a **tar.gz** file, you first must copy that file to the build tree's **SOURCES** directory.

Directory Name	Description
BUILD	The directory where RPM does all its building
SOURCES	The directory where you should put your original source archive files and your patches
SPECS	The directory where all spec files should go
RPMS	The directory where RPM puts all binary RPMs when built
SRPMS	The directory where all source RPMs are put

The following example copies the compressed archive for the bookrec software to the **SOURCES** directory:

```
# cp bookrec-1.0.tar.gz  /usr/src/redhat/SOURCES
```

The **topdir:** entry in an **rpmrc** file, like **/usr/lib/rpm/rpmrc**, specifies the location of the build tree directories. In this file, you can find an entry for **topdir:**. Currently, the Red Hat system has already set this directory to **/usr/src/redhat**. You can find the **SOURCES**, **BUILD**, **RPMS**, and **SRPMS** directories here. You can specify a different directory for these subdirectories by placing the entry for **topdir:** in the **/etc/rpmrc** file.

```
topdir: /usr/src/redhat
```

A sample of the default values set by **/usr/lib/rpm/rpmrc** is shown here:

```
# Default values, often overridden in /etc/rpmrc
dbpath:       /var/lib/rpm
topdir:      /usr/src/redhat
tmppath:     /var/tmp
cpiobin:     cpio
defaultdocdir:   /usr/doc
```

By default, RPM is designed to work with source code placed in a directory consisting of its name and a release number, separated by a hyphen. For example, a program with the name **bookrec** and **release 1.0** should have its source-code files in a directory called **bookrec-1.0**. If RPM needs to compile the software, it expects to find the source code in that directory within the **BUILD** directory, **BUILD/bookrec-1.0**. The same name and release number also must be specified in the spec file.

RPM Spec File

To create a package, first create an **rpm** spec file for it. The **rpm** spec file specifies the files to be included, any actions to build the software, and information about the package. The spec file is designed to take the program through several stages, starting with unpacking it from an archive, compiling its source code, and generating the RPM package. In the spec file are segments for the different stages and special RPM macros that perform actions at these stages. These are listed here:

File Segment or Macro	Description
%description	A detailed description of the software.

File Segment or Macro	Description
%prep	The prep stage for archives and patches.
%setup	The prep macro for unpacking archives.A -n *name* option resets the name of the build directory.
%patch	The prep macro for updating patches.
%build	The build stage for compiling software.
%install	The install stage for installing software.
%files	The files stage that lists the files to be included in the package.A -f *filename* option specifies a file that contains a list of files to be included in the package.
%config *file-list*	A file macro that lists configuration files to be placed in the /etc directory.
%doc *file-list*	A file macro that lists documentation files to be placed in the /usr/doc directory with the subdirectory of the name-version-release.
%dir *directory-list*	The specification of a directory to be included as being owned by a package. (A directory in a file list refers to all files in it, not only the directory.)
%pre	A macro to do preinstall scripts.
%preun	A macro to do preuninstall scripts.
%post	A macro to do postinstall scripts.
%postun	A macro to do postuninstall scripts.

A spec file is divided into five basic segments: header, prep, build, install, and files. These segments are separated in the file by empty lines. The header segment contains several lines of information, each preceded by a tag and a semicolon. For example, the following tag is used for a short description of the software:

```
Summary: bookrec program to manage book records
```

The name, version, and release tags are used to build the name of the RPM package. The name, version, and release are separated with hyphens. For example, the name **bookrec** with the **version 1.0** and **release 2** has the following name:

```
bookrec-1.0-2
```

The Group entry is a list of categories for the software and is used by the RPM package management utilities like GnomeRPM and Linuxconf to place the software in the correct category folder. The Source entry is the compressed archive where the software is stored on your system. Description is a detailed description of the software.

Following the header are the three stages for creating and installing the software on your system, indicated by the **%prep**, **%build**, and **%install rpm** macros. You can skip any of these stages, say, if the software is already installed. You can also leave any of them out of the spec file or comment them out with a preceding #. The spec file is capable of taking a compressed archive, unpacking it, compiling the source code files, and then installing the program on your system. Then the installed files can be used to create the RPM package.

The **%prep** macro begins the prep segment of the spec file. The prep segment's task is to generate the software's source code. This usually means unpacking archives, but it may also have to update the software with patches. The tasks themselves can be performed by shell scripts you write. Special macros can also automatically perform these tasks. The **%setup** macro can decompress and unpack an archive in the **SOURCES** directory, placing the source code files in the **BUILD** directory. The **%patch** macro applies any patches.

The **%build** segment contains the instructions for compiling the software. Usually, this is a simple **make** command, depending on the complexity of your program. The **%install** segment contains the instructions for installing the program. You can use simple shell commands to copy the files or, as in the **bookspec** example that follows, the **install** command that installs files on systems. This could also be the **make install** command, if your **Makefile** has the commands to install your program.

```
%build
make RPM_OPT_FLAGS="$RPM_OPT_FLAGS"

%install
install -s -m 755 -o 0 -g 0 bookrec /usr/bin/bookrec
install -m 644 -o 0 -g 0 bookrec.1 /usr/man/man1
```

The **%files** segment contains the list of files you want placed in the RPM package. Following the **%files** macro, you list the different files, including their full pathnames. The macro **%config** can be used to list configuration files. Any files listed here are placed in the **/etc** directory. The **%doc** macro is used for documentation, such as **README** files. These are placed in the **/usr/doc** directory under a subdirectory consisting of the software's name, version, and release number. In the **bookspec** example shown here, the **readme** file is placed in the **/usr/doc/bookrec-1.0-2** directory:

bookspec

```
Summary: bookrec program to manage book records
Name: bookrec
Version: 1.0
Release: 2
Copyright: GPL
Group: Applications/Database
Source: /root/rpmc/bookrec-1.0.tar.gz
%description
This program manages book records by title, providing
price information

%prep
%setup

%build
make RPM_OPT_FLAGS="$RPM_OPT_FLAGS"

%install
install -s -m 755 -o 0 -g 0 bookrec /usr/bin/bookrec
install -m 644 -o 0 -g 0 bookrec.1 /usr/man/man1
```

```
%files
%doc README

/usr/bin/bookrec
/usr/man/man1/bookrec.1
```

RPM Build Operation

To create an RPM software package, you use the rpm build options (listed in Table 31-4) with the **rpm** command, followed by the name of a spec file. The **-bl** option checks to see if all the files used for the software are present. The **-bb** option builds only the binary package, whereas **-ba** builds both binary and source packages. They expect to find the compressed archive for the software in the build tree's **SOURCES** directory. The **-ba** and **-bb** options execute every stage specified in the **rpm** spec script, starting from the prep stage, to unpacking an archive, and then compiling the program, followed by installation on the system, and then creation of the package. The completed RPM package for executable binaries is placed in a subdirectory of the build tree's **RPMS** directory. This subdirectory has a name representing the current platform. For a PC, this is **i386**, and the package is placed in the **RPMS/i386** subdirectory. The source-code package is placed directly in the **SRPMS** directory.

Table 31-4: The RPM Build Options

Option	**Description**
-ba	Create both the executable binary and source code packages. Perform all stages in the spec file: prep, build, install, and create the packages.
-bb	Create only the executable binary package. Perform all stages in the spec file: prep, build, install, and create the package.
-bp	Run only the prep stage from the spec file (**%prep**).
-bl	Do a "list check." The **%files** section from the spec file is macro-expanded, and checks are made to ensure the files exist.
-bc	Do both the prep and build stages, unpacking and compiling the software **(%prep** and **%build**).
-bi	Do the prep, build, and install stages, unpacking, compiling, and installing the software (**%prep**, **%build**, and **%install**).
--short-circuit	Skip to specified stage, not executing any previous stages. Only valid with **-bc** and **-bi**.
--clean	Remove the build tree after the packages are made.
--test	Do not execute any build stages. Used to test spec files.
--recompile *source_package_file*	RPM installs the source code package and performs a prep, compile, and install.
--rebuild *source_package_file*	RPM first installs the named source package and does a prep, compile, and install, and then rebuilds a new binary package.
--showrc	List the configuration variables for the **/usr/lib/rpm/rpmrc** file.

The following program generates both a binary and a software package, placing them in the build tree's **RPMS/i386** and **SRPMS** directories. The name of the spec file in this example is **bookspec**.

```
rpm -ba bookspec
```

An executable binary package has a name consisting of the software name, the version number, the release number, the platform name (i386), and the term "rpm". The name, version, and release are separated by hyphens, whereas the release, platform name, and the rpm term are separated by periods. The name of the binary package generated by the previous example, using the **bookspec** spec script, generates the following name:

```
bookrec-1.0-2.i386.rpm
```

The source code package has the same name, but with the term "src" in place of the platform name:

```
bookrec-1.0-2.src.rpm
```

Chapter 32: File System Administration

Overview

Files reside on physical storage devices such as hard drives, CD-ROMs, or floppy disks. The files on each storage device are organized into a file system. The storage devices on your Linux system are treated as a collection of file systems that you can manage. When you want to add a new storage device, you will need to format it as a file system and then attach it to your Linux file structure. Hard drives can be divided into separate storage devices called *partitions*, each of which would have its own file system. You can perform administrative tasks on your file systems, such as backing them up, attaching or detaching them from your file structure, formatting new devices or erasing old ones, and checking a file system for problems. This chapter discusses how you can manage file systems on your storage devices such as CD-ROMs, floppy disks, and hard disk partitions.

To access files on a device, you attach its file system to a specified directory. This is called *mounting* the file system. For example, to access files on a floppy disk, you first mount its file system to a particular directory. With Linux, you can mount a number of different types of file systems. You can even access a Windows hard drive partition or tape drive, as well as file systems on a remote server (see Chapter 37).

Archives are used to back up files or to combine them into a package, which can then be transferred as one file over the Internet or posted on an FTP site for easy downloading. The standard archive utility used on Linux and Unix systems is tar, for which several GUI front ends exist. You have several compression programs to choose from, including GNU zip (gzip), Zip**,** bzip**,** and compress.

Local File Systems

Your Linux system is capable of handling any number of storage devices that may be connected to it. You can configure your system to access multiple hard drives, partitions on a hard drive, CD-ROM disks, floppy disks, and even tapes. You can elect to attach these storage components manually or have them automatically mount when you boot. For example, the main partition holding your Linux system programs is automatically attached whenever you boot, whereas a floppy disk must be manually attached when you put one in your floppy drive. You can configure this access to different storage devices either by manually editing configuration files, such as **/etc/fstab**, or by using a file system configuration tool such as the Linuxconf's fsconf. You can use administration tools such as Linuxconf or Webmin to configure the file system.

File Systems

Although all the files in your Linux system are connected into one overall directory tree, the files themselves reside on storage devices such as hard drives or CD-ROMs. The Linux files on a particular storage device are organized into what is referred to as a *file system.* Your Linux directory tree may encompass several file systems, each on different storage devices. On a hard drive with several partitions, you would have a file system for each partition. The files themselves are organized into one seamless tree of directories, beginning from the root directory. Although the root may be located in a file system on a hard drive partition, a pathname leads directly to files located on the file system for your CD-ROM.

The files in a file system remain separate from your directory tree until you specifically connect them to it. A file system has its files organized into its own directory tree. You can think of this as a *subtree* that must be attached to the main directory tree. For example, a floppy disk with Linux files has its own tree of directories. You need to attach this subtree to the main tree on your hard drive partition. Until they are attached, you cannot access the files on your floppy disk.

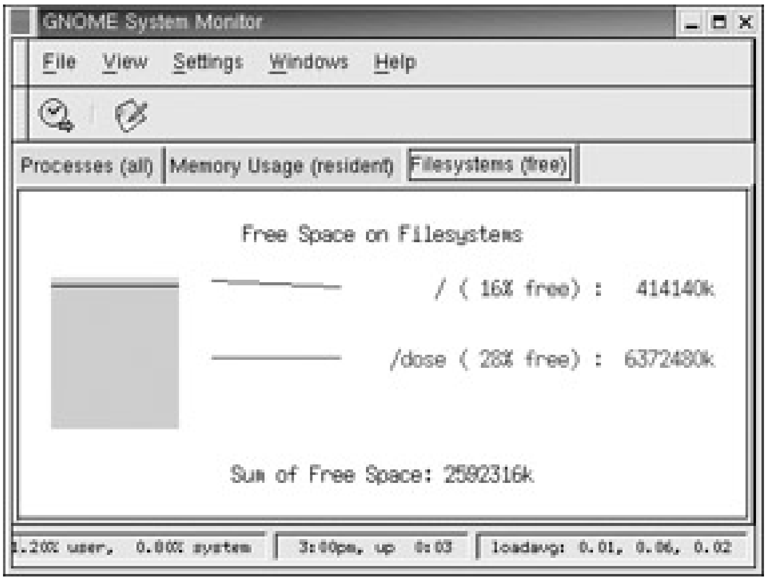
Attaching a file system on a storage device to your main directory tree is called *mounting the device.* The **mount** operation attaches the directory tree on the storage device to a directory you specify. You can then change to that directory and access those files. The directory in the file structure to which the new file system is attached is referred to as the *mountpoint.* For example, to access files on a CD-ROM, first you have to mount the CD-ROM.

Currently, Linux systems have several ways to mount a file system. You can use Linuxconf to select and mount a file system easily. If you are using either Gnome or the K Desktop, you can use special desktop icons to mount a file system. From a shell command line, you can use the **mount** command. Mounting file systems can only be done as the root user. This is a system administration task and cannot be performed by a regular user. To mount a file system, be sure to log in as the root user (or use the **su** operation). As the root user, you can, however, make a particular device like a CD-ROM user mountable. In this way, any user could put in a CD-ROM and mount it. You could do the same for a floppy drive.

Tip On Gnome you can use the Disk Management tool on the System menu to mount and unmount file systems, including floppy disks and CD-ROMs. On KDE you can use the KDiskFree utility, which also lists your mountable file as well as their disk usage.

For a file system to be accessible, it must be mounted. Even the file system on your hard disk partition must be mounted with a **mount** command. When you install your Linux system and create the Linux partition on your hard drive, however, your system is automatically configured to mount your main file system whenever it starts. Floppy disks and CD-ROMs must be explicitly mounted. Remember, when you mount a CD-ROM or floppy disk, you cannot then simply remove it to put in another one. You first have to unmount it. In fact, the CD-ROM drive remains locked until you unmount it. Once you unmount a CD-ROM, you can then take it out and put in another one, which you then must mount before you can access it. When changing several CD-ROMs or floppy disks, you are continually mounting and unmounting them.

The file systems on each storage device are formatted to take up a specified amount of space. For example, you may have formatted your hard drive partition to take up 3GB. Files installed or created on that file system take up part of the space, while the remainder is available for new files and directories. To find out how much space you have free on a file system, you can use the **df** command or, on Gnome, you can use either the Gnome System Monitor (see Figure 32-1) or the Gnome Disk Free utility. For the Gnome System Manger, click the Filesystems tab to display a bar graph of the free space on your file systems. Gnome DiskFree displays a list of meters showing how much space is used on each partition and how much space you have left. KDiskFree, a KDE utility, provides similar information.



```
Free Space on Filesystems
/ ( 16% free) :  414140k
/dose ( 28% free) :  6372480k
Sum of Free Space: 2582316k
```

```
1.20% user,  0.00% system | 3:00pm, up  0:03 | loadavg: 0.01, 0.06, 0.02
```

Figure 32-1: Gnome System Monitor, Filesystems tab

The **df** command lists all your file systems by their device names, how much memory they take up, and the percentage of the memory used, as well as where they are mounted. With the **-h** option, it displays information in a more readable format. The **df** command is also a safe way to obtain a listing of all your partitions, instead of using **fdisk**. **df** only shows mounted partitions, however, whereas **fdisk** shows all partitions.

```
$ df
Filesystem 1024-blocks Used Available Capacity Mounted on
/dev/hda3 297635 169499 112764 60% /
/dev/hda1 205380 182320 23060 89% /mnt/dos
/dev/hdc 637986 637986 0 100% /mnt/cdrom
```

Note In earlier Red Hat versions, fsck was also used to recover file systems after disk crashes or reset-button reboots. With release 7.2, Red Hat introduced journaling capabilities with the ext3 file system. Journaling provides for fast and effective recovery in case of disk crashes, instead of using fsck or e2fsck.

You can also use **df** to tell you to what file system a given directory belongs. Enter **df** with the directory name or **df.** for the current directory.

```
$ df .
Filesystem 1024-blocks Used Available Capacity Mounted on
/dev/hda3 297635 169499 112764 60% /
```

To make sure nothing is wrong with a given file system, you can use the **fsck** command to check it. However, be sure that the file system is unmounted. **fsck** should not be used on a mounted file system. To use **fsck**, enter **fsck** and the device name that references the file system. **fsck** is run automatically on all your file systems when you boot up your system, so your file systems are continually checked. Table 32-1 lists the **fsck** options. The following examples check the disk in the floppy drive and the primary hard drive:

```
# fsck /dev/fd0
# fsck /dev/hda1
```

Table 32-1: Thefsck Options for Checking and Repairing File Systems

Option	Description
file-system	Specifies the file system to be checked. Use file system's device name, such as /**dev/hda3**.
-A	Checks all file systems listed in /**etc/fstab** file.
-V	Verbose mode. List actions that **fsck** takes.
-t *file-system-type*	Specifies the type of file system to be checked.
-a	Automatically repairs any problems.
-l	Lists the names of all files in the file system.
-r	Asks for confirmation before repairing file system.
-s	Lists superblock before checking file system.

Note Instead of using **fsck**, you can use **fse2ck** to check standard Linux partitions (ext2).

Filesystem Hierarchy Standard

Linux organizes its files and directories into one overall interconnected tree, beginning from the root directory and extending down to system and user directories (see Chapter 11). The organization and layout for the system directories is determined by the Filesystem Hierarchy Standard (FHS). The FHS provides a standardized layout that all Linux distributions should following setting up their system directories. For example, there must be an /**etc** directory to hold configuration files and a /**dev** directory for device files. You can find out more about FHS, including the official documentation, at **www.pathname.com/fhs**. The current release is FHS 2.1, which is the successor to FSSTND 1.2, a precursor to FHS. Linux distributions, developers, and administrators all follow the FHS to provide a consistent organization to the Linux file system.

Linux uses a number of specifically named directories for specialized administration tasks. All these directories are directories at the very top level of your main Linux file system, the file system root directory represented by a single slash, /. For example, the /**dev** directory holds device files, the **/etc** directory holds configuration files, and the **/home** directory holds the user home directories and all their user files. You only have access to these directories and files as the system administrator. You need to log in as the root user, placing you in a special root user administrative directory called **/root**. From here, you can access any directory on the Linux file system, both administrative and user.

The directories held in the root directory, /, are listed in Table 32-2, along with other useful subdirectories. Ones that you may commonly access as an administrator are the **/etc** directory that holds configuration files, the **/dev** directory that holds device files, and the **/var** directory that holds server data files for DNS, Web, mail, and FTP servers along with system logs and scheduled tasks. For managing different versions of the kernel, you may need to access the **/boot** and **/lib/modules** directories. The **/boot** directory will hold the kernel image files for any new kernels you install, and the **/lib/modules** directory will hold modules for your different kernels.

Table 32-2: Linux File System Directories

Directory	**Function**
/	Begins the file system structure-called the root.
/boot	Holds the kernel image files and modules loaded when your system boots up.
/home	Contains users' home directories.
/sbin	Holds administration level commands and any used by the root user.
/dev	Holds file interfaces for devices such as the terminal and printer.
/etc	Holds system configuration files and any other system files.
/etc/opt	Holds system configuration files for applications in **/opt**.
/etc/X11	Holds system configuration files for the X Window System and its applications.
/bin	Holds the essential user commands and utility programs.
/lib	Holds essential shared libraries and kernel modules.
/lib/modules	Holds the kernel modules.
/mnt	Used to hold directories for mounting file systems like CD-ROMs or floppy disks that are mounted only temporarily.
/opt	Holds added software applications (for example, KDE on some distributions).
/proc	Process directory, a memory-resident directory containing files used to provide information about the system.
/tmp	Holds temporary files.
/usr	Holds those files and commands used by the system; this directory breaks down into several subdirectories.
/var	Holds files that vary, such as mailbox files. On Red Hat, these also hold Web and FTP server data files.

The **/usr** directory contains a multitude of important subdirectories used to support users, providing applications, libraries, and documentation (see Table 32-3). **/usr/bin** holds numerous user-accessible applications and utilities. **/usr/sbin** hold user-accessible administrative utilities. The **/usr/share** directory holds architecture-independent data that includes an extensive number of subdirectories, including those for the documentation such as Man, info, and doc files.

Table 32-3: /usr Directories

Directory	**Description**
/usr/bin	Holds most user commands and utility programs.
/usr/sbin	Holds nonessential administrative applications.
/usr/lib	Holds libraries for applications, programming languages, desktops, etc.
/usr/games	Games and educational programs.
/usr/include	C programming language header files (**.h**).
/usr/doc	Holds Linux documentation.
/usr/local	Directory for locally installed software.
/usr/share	Architecture independent data such as documentation like Man and info pages.
/usr/src	Holds source code, including the kernel source codes.
/usr/X11R6	X Window System-based applications and libraries.

The **/var** directories are designed to hold data that changes with the normal operation of the Linux system (see Table 32-4). For example, spool files for documents that you are printing are kept here. A spool file is created as a temporary printing file and is removed after printing. Other files, like system log files, are changed constantly.

Table 32-4: /var Directories

Directory	**Description**
/var/account	Processes accounting logs.
/var/cache	Application cache data for Man pages, Web proxy data, fonts, or application-specific data.
/var/crash	System crash dumps.
/var/games	Varying games data.
/var/lib	Holds state information for particular applications.
/var/local	Used for data that changes for programs installed in **/usr/local**.
/var/lock	Holds lock files that indicate when a particular program or file is in use.
/var/log	Holds log files such as **/var/log/messages** that contain all kernel and system program messages.
/var/mail	User mailbox files.
/var/opt	Variable data for applications installed in **/opt**.
/var/run	Information about system's running processes.
/var/spool	Holds application's spool data such as that for mail, news, and

Table 32-4: /var Directories

Directory	Description
	printer queues as well as **cron** and **at** jobs.
/var/tmp	Holds temporary files that should be preserved between system reboots.
/var/yp	Network Information Service (NIS) data files.

The **/proc** file system is a special file system that is generated in system memory (see Table 32-5). It does not exist on any disk. **/proc** contains files that provide important information about the state of your system. For example, **/proc/cpuinfo** holds information about your computer's CPU processor. **/proc/devices** will list those devices currently configured to run with your kernel. **/proc/filesystems** will list the file systems. **/proc** files are really interfaces to the kernel, obtaining information from the kernel about your system.

Table 32-5: /proc Directories and Files

File	Description
/proc/*num*	There is a directory for each process labeled by its number. **/proc/1** is the directory for process 1.
/proc/cpuinfo	Contains information about the CPU, such as its type, make, model, and performance.
/proc/devices	List of the device drivers configured for the currently running kernel.
/proc/dma	Displays the DMA channels currently used.
/proc/filesystems	File systems configured into the kernel.
/proc/interrupts	Displays the interrupts in use.
/proc/ioports	Shows the I/O ports in use.
/proc/kcore	Holds an image of the physical memory of the system.
/proc/kmsg	Messages generated by the kernel.
/proc/ksyms	Symbol table for the kernel.
/proc/loadavg	The system "load average."
/proc/meminfo	Memory usage.
/proc/modules	Lists the kernel modules currently loaded.
/proc/net	Status information about network protocols.
/proc/stat	System operating statistics, such as page fault occurrences.
/proc/uptime	The time the system has been up.
/proc/version	The kernel version.

Device Files: /dev

To mount a file system, you have to specify its device name. The interfaces to devices that may be attached to your system are provided by special files known as *device* files. The names of these device files are the device names. Device files are located in the **/dev** directories and usually have abbreviated names ending with the number of the device. For example, **fd0** may reference the first floppy drive attached to your system. On Linux systems operating on PCs, the IDE hard disk partitions have a prefix of **hd,** followed by an alphabetic character that labels the hard drive, and then a number for the partition. For example, **hda2** references the second partition on the first IDE hard drive. The prefix **sd** references SCSI hard drives, so **sda3** would reference the second partition on the first SCSI hard drive. In most cases, you can use the **man** command with a prefix to obtain more detailed information about this kind of device. For example, **man sd** displays the Man pages for SCSI devices. A complete listing of all device names can be found in the **devices** file located in the **linux/doc/device-list** directory at the **www.kernel.org** Web site. Table 32-6 lists several of the commonly used device names.

Table 32-6: Device Name Prefixes

Device Name	**Description**
hd	IDE hard drives 1-4 are primary partitions, and 5 and up are logical partitions
sd	SCSI hard drives
scd	SCSI CD-ROM drives (used on Red Hat)
sr	SCSI CD-ROM drives (alternative prefix name, may be used on other distributions)
fd	Floppy disks
st	SCSI tape drives
nst	SCSI tape drives, no rewind
ht	IDE tape drives
tty	Terminals
lp	Printer ports
pty	Pseudoterminals (used for remote logins)
js	Analog joy sticks
midi	Midi ports
ttyS	Serial ports
md	RAID devices
rd/c*n*d*n*	Directory that holds RAID devices is **rd**. **c*n*** is the RAID controller and **d*n*** is the RAID disk for that controller
cdrom	Link to your CD-ROM device file
cdwriter	Link to your CD-R or CD-RW device file
modem	Link to your modem device file
floppy	Link to your floppy device file
tape	Link to your tape device file
scanner	Link to your scanner device file

The device name for your floppy drive is **fd0** and is located in the directory **/dev**. **/dev/fd0** references your floppy drive. Notice the numeral **0** after **fd**. If you have more than one floppy drive, they are represented by **fd1**, **fd2**, and so on.

IDE hard drives use the prefix **hd**, while SCSI hard drives use the prefix **sd**. RAID devices, on the other hand, will use the prefix **md**. The prefix for a hard disk is followed by an alphabetic character that labels the hard drive and a number for the partition. For example, **hda2** references the second partition on the first IDE hard drive, where the first hard drive is referenced with the character "a", as in **hda**. The device **sdb3** refers to the third partition on the second SCSI hard drive (**sdb**). RAID devices, however, are numbered from 0, like floppy drives. Device **md0** references the first RAID device and **md1**, the second. To find the device name, you can use **df** to display your hard partitions or examine the **/etc/fstab** file.

The device name for your CD-ROM drive varies depending on the type of CD-ROM you have. The device name for an IDE CD-ROM has the same prefix as an IDE hard disk partition, **hd**, and is identified by a following character that distinguishes it from other IDE devices. For example, an IDE CD-ROM connected to your secondary IDE port may have the name **hdc**. An IDE CD-ROM connected as a slave to the secondary port may have the name **hdd**. The actual name is determined when the CD-ROM is installed, as happened when you installed your Linux system. SCSI CD-ROM drives use a different nomenclature for their device names. They begin with **scd** for SCSI drive and are followed by a distinguishing number. For example, the name of an SCSI CD-ROM could be **scd0** or **scd1**. The name of your CD-ROM was determined when you installed your system. You can find out what it is either by examining the **/etc/fstab** file or using Linuxconf on your root user desktop.

Mount Configuration: /etc/fstab

Although you can mount a file system directly with only a **mount** command, you can simplify the process by placing mount information in the **/etc/fstab** configuration file. Using entries in this file, you can have certain file systems automatically mounted whenever your system boots. For others, you can specify configuration information, such as mountpoints and access permissions, which can be automatically used whenever you mount a file system. You needn't enter this information as arguments to a **mount** command as you otherwise must. This feature is what allows mount utilities on Gnome, KDE, and Linuxconf to enable you to mount a file system simply by clicking a button. All the mount information is already in the **/etc/fstab** file. For example, when adding a new hard disk partition to your Linux system, you most likely want to have it automatically mounted on startup, and then unmounted when you shut down. Otherwise, you must mount and unmount the partition explicitly each time you boot up and shut down your system. To have Linux automatically mount the file system on your new hard disk partition, you only need to add its name to the **fstab** file. You can do this by directly and carefully editing the **/etc/fstab** file to type in a new entry, or you can use Linuxconf as described in the next section.

An entry in an **fstab** file contains several fields, each separated by a space or tab. These are described in Red Hat as the device, mountpoint, file system type, options, dump, and fsck fields, arranged in the sequence shown here:

```
<device> <mountpoint> <filesystemtype> <options> <dump> <fsck>
```

The first field is the name of the file system to be mounted. This usually begins with **/dev**, such as **/dev/hda3** for the third hard disk partition. The next field is the directory in your file structure where you want the file system on this device to be attached. The third field is the type of file system being mounted. Table 32-7 provides a list of all the different types you can mount. The type for a standard Linux hard disk partition is **ext2**. The next example shows an entry for the main Linux hard disk partition. This entry is mounted at the root directory, /, and has a file type of **ext2**.

Table 32-7: File System Types

Type	**Description**
auto	Attempt to automatically detect the file system type
minux	Minux file systems (filenames are limited to 30 characters)
ext	Earlier version of Linux file system, no longer in use
ext2	Standard Linux file system supporting large filenames and file sizes; does not include journaling
ext3	Standard Linux file system supporting large filenames and file sizes; includes journaling
xiaf	Xiaf file system
msdos	File system for MS-DOS partitions (16-bit)
vfat	File system for Windows 95, 98, and Millennium partitions (32-bit)
ntfs	NT and Windows 2000 file systems
hpfs	File system for OS/2 high-performance partitions
proc	Used by operating system for processes
nfs	NFS file system for mounting partitions from remote systems
umsdos	UMS-DOS file system
swap	Linux swap partition or swap file
sysv	UNIX System V file systems
iso9660	File system for mounting CD-ROM

```
/dev/hda3 / ext2 defaults 0 1
```

The type of file system on a floppy drive could vary, often depending on the type floppy you are trying to mount. For example, you may want to read a Windows formatted floppy disk at one time and a Linux formatted floppy disk at another time. For this reason, the file system type specified for the floppy device is **auto**. With this option, the type of file system formatted on the floppy disk will be automatically detected and the appropriate file system type used.

```
/dev/fd0  /mnt/floppy  auto   defaults,noauto   0 0
```

The field after the file system type lists the different options for mounting the file system. You can specify a default set of options by simply entering **defaults**. You can list specific options next to each other separated by a comma (no spaces). The **default**s option specifies that a device is read/write (**rw**), asynchronous (**async**), a block device (**dev**), cannot be mounted by ordinary users (**user**) , and that programs can be executed on it (**exec**). By contrast, a CD-ROM only has a few options listed for it: **ro** and **noauto**. **ro** specifies this is read-only; and **noauto** specifies this is not automatically mounted. The **noauto** option is used with both CD-

ROMs and floppy drives, so they won't automatically mount because you do not know if you have anything in them when you start up. At the same time, the entries for both the CD-ROM and the floppy drive specify where they are to be mounted when you decide to mount them. Table 32-8 lists the options for mounting a file system. The floppy drive entry also has all the **default** options of the hard disk partitions, with the exception that it is not automatically mounted (not mountable with the **-a** option). An example of CD-ROM and floppy drive entries follows. Notice the type for a CD-ROM file system is different from a hard disk partition, **iso9660**.

```
/dev/hdc /mnt/cdrom   iso9660 ro,noauto          0 0
/dev/fd0 /mnt/floppy  auto    defaults,noauto    0 0
```

Table 32-8: Mount Options for File Systems:-o and /etc/fstab

Option	**Description**
async	All I/O to the file system should be done asynchronously.
auto	Can be mounted with the **-a** option. A **mount -a** command executed when the system boots, in effect, mounts file systems automatically.
defaults	Use default options: **rw**, **suid**, **dev**, **exec**, **auto**, **nouser**, and **async**.
dev	Interpret character or block special devices on the file system.
noauto	Can only be mounted explicitly. The **-a** option does not cause the file system to be mounted.
exec	Permit execution of binaries.
nouser	Forbid an ordinary (that is, nonroot) user to mount the file system.
remount	Attempt to remount an already mounted file system. This is commonly used to change the mount flags for a file system, especially to make a read-only file system writable.
ro	Mount the file system read-only.
rw	Mount the file system read/write.
suid	Allow set-user-identifier or set-group-identifier bits to take effect.
sync	All I/O to the file system should be done synchronously.
user	Enable an ordinary user to mount the file system. Ordinary users always have the following options activated: **noexec**, **nosuid**, and **nodev**.
nodev	Do not interpret character or block special devices on the file system.
nosuid	Do not allow set-user-identifier or set-group-identifier bits to take effect.

The last two fields consist of an integer value. The first one is used by the **dump** command to determine if a file system needs to be dumped, backing up the file system. The last one is used by **fsck** to see if a file system should be checked at reboot and in what order. If the field has a value of 1, it indicates a boot partition, and 2 indicates other partitions. The 0 value means the **fsck** needn't check the file system.

A copy of an **/etc/fstab** file is shown here. Notice the first line is comment. All comment lines begin with a #. The entry for the **/proc** file system is a special entry used by your Linux operating system for managing its processes, and it is not an actual device. To make an entry in the **/etc/fstab** file, you can either edit the **/etc/fstab** file directly or use Linuxconf, which prompts you for information and then makes the correct entries into your **/etc/fstab** file. You can use the **/etc/fstab** example here as a guide to show how your entries should look. The **/proc** and **swap** partition entries are particularly critical.

/etc/fstab

```
# <device> <mountpoint> <filesystemtype> <options> <dump><fsck>
/dev/hda3  /            ext3             defaults  0     1
/dev/hdc   /mnt/cdrom   iso9660          ro,noauto 0     0
/dev/fd0   /mnt/floppy  auto             defaults,
                                         noauto    0
/proc      /proc        proc             defaults
/dev/hda2  none         swap             sw
/dev/hda1  /mnt/windows vfat             defaults  0     0
```

You can mount either MS-DOS or Windows 95 partitions used by your MS-DOS or Windows operating system onto your Linux file structure, just as you would mount any Linux file system. You only have to specify the file type of **vfat** for Windows 95 and **msdos** for MS-DOS. You may find it convenient to have your Windows partitions automatically mounted when you start up your Linux system (the same is true for MS-DOS partitions). To do this, you need to put an entry for your Windows partitions in your **/etc/fstab** file and give it the **defaults** option or be sure to include an **auto** option. You make an entry for each Windows partition you want to mount, and then specify the device name for that partition followed by the directory in which you want to mount it. The **/mnt/windows** directory would be a logical choice (be sure the **windows** directory has already been created in **/mnt**). For the file system type, enter **vfat**. The next example shows a standard MS-DOS partition entry for an **/etc/fstab** file. Notice the last entry in the **/etc/fstab** file example was an entry for mounting a Windows partition.

```
/dev/hda1 /mnt/windows vfat defaults 0 0
```

If your **/etc/fstab** file ever becomes corrupt-say, a line gets deleted accidentally or changed-then your system will boot into a maintenance mode, giving you read-only access to your partitions. To gain read/write access so you can fix your **/etc/fstab** file, you have to remount your main partition. The following command performs such an operation:

```
# mount -n -o remount,rw /
```

File systems listed in the **/etc/fstab** file are automatically mounted whenever you boot, unless this feature is explicitly turned off with the **noauto** option. Notice the CD-ROM and floppy disks have a **noauto** option. Also, if you issue a **mount -a** command, all the file systems without a **noauto** option are mounted. If you would want to make the CD-ROM user mountable, add the **user** option.

```
/dev/hdc /mnt/cdrom iso9660 ro,noauto,user 0 0
```

Note The "automatic" mounting of file systems from **/etc/fstab** is actually implemented by executing a **mount -a** command in the **/etc/rc.d/rc.sysinit** file that is run whenever you boot. The **mount -a** command will mount any file system listed in your **/etc/fstab** file that does **not** have a **noauto** option. The **umount -a** option will unmount the file systems in **/etc/fstab** (which is executed when you shut down your system).

Installing IDE CD-R and CD-RW Devices

Linux CD Writing applications all treat CD-R and CD-RW drives as if they were SCSI drives. This means that IDE CD-R and CD-RW drives have to emulate SCSI drives for them to be recognized and used by CD Writing software. Even if you want to use an IDE CD-ROM in a CD writing application, say as just the reader to copy a CD disk, that IDE CD-ROM drive would still have to emulate a SCSI CD-ROM drive. Only SCSI drives (CD-R, CD-RW, or CD-ROM) are recognized by Linux CD Writers. For example, if you have a regular IDE CD-ROM and you want to use it in Linux CD-write software to copy CDs (ripping), then you still have to have that IDE CD-ROM emulate an SCSI CD-ROM. Check the CD-Writing HOWTO at **www.linuxdoc.org** for more details. A brief description is shown here.

Note SCSI emulation for IDE devices is implemented in the kernel as "SCSI Emulation Support" in the "IDE, ATA, and ATAPI Block Devices entry," located in the "ATA/ IDE/MFM/RLL Support" window opened from the main kernel configuration menu. Normally it is compiled as a module.

IDE CD drives (CD-R, CD-RW, and CD-ROM) will be recognized as IDE devices during installation and installed as such. However, when you start up your system, you need to instruct the Linux kernel to have the IDE CD drives emulate SCSI CD drives. This means that a CD-R drive that would be normally recognized as a **/dev/hdc** drive would have to be recognized as a **/dev/scd0** device, the first SCSI CD-ROM drive. You do this by loading the **ide-scsi** module, which allows an IDE CD drive to emulate an SCSI CD drive.

You can implement SCSI emulation for IDE CD drives in one of two ways: either by loading the **ide-scsi** module as a kernel parameter, or specifying the module in the **/etc/modules.conf** file. You will also have to indicate the IDE drives that will be emulated. If the **ide-scsi** module is compiled into the kernel (not as a separate module), then you have to load it as a kernel parameter. An **ide-scsi** module can be loaded either way.

IDE SCSI Emulation in Kernel Parameters

To create a kernel parameter, you can either manually enter the **ide-scsi** parameter at the boot prompt or place it in the boot loader file (**/etc/lilo.conf**) file to have it automatically entered. The parameter is read when the kernel boots. List each IDE CD-R or CD-RW that needs to emulate a SCSI CD-R or CD-RW as using the **ide-scsi** module. You assign the **ide-scsi** module to the device name of the IDE CD drive to be emulated. The following example would load the **ide-scsi** module to have the master IDE drive on the secondary IDE connection (hdc) emulate a SCSI drive.

```
hdc=ide-scsi
```

The following example shows how two IDE CD drives are specified at the Linux boot prompt.

```
boot:  linux hdc=ide-scsi hdd=ide-scsi
```

For the **/etc/lilo.conf** file, you assign these parameters to an **append** command in the image segment for the Linux kernel. The **append** command is assigned a string listing the parameters for that kernel image. Be sure to enclose the string in double quotes. The following example shows the kernel paramater that configures the hdc IDE drive to emulate a SCSI drive, enabling it to operate as a CD-R or CD-RW drive. Arguments like this are used by GRUB in its grub.conf file.

```
append="hdc=ide-scsi"
```

For two IDE CD drives, you would use both parameters in the same string.

```
append="hdc=ide-scsi hdd=ide-scsi"
```

Be sure to execute the **lilo** command to effect the changes you make to **/etc/lilo.conf**.

IDE SCSI Emulation in /etc/modules.conf

If you are not using LILO or, for some reason, do not want to modify the **/etc/lilo.conf** file, you can instead configure the **/etc/modules.conf** file to load and implement the SCSI emulation for your IDE CD drives. This involves entering several module configuration commands in the **/etc/modules.conf** file. When your system starts up, it will load the modules as specified in that file.

Initially, your IDE CD drives will be recognized and configured as IDE CD-ROMs by the **ide-cd** module. You will need to specify options for the **ide-cd** module to ignore the IDE CD-R and CD-RW drives, as well as any CD-ROM drives you want to use in CD Writing applications. You do this with the **options** command and the **ignore** option. The following example instructs **ide-cd** to ignore an IDE CD-R that has been installed as an IDE CD-ROM at **/dev/hdc**, the secondary IDE master.

```
options ide-cd ignore=hdc
```

You then have to enter **alias** commands to identify the scd SCSI drives as using the SCSI CD module, **sr_mod**. The following example aliases the **/dev/scd0** device as using the **sr_mod** SCSI CD module.

```
alias scd0 sr_mod
```

There are also several pre-install commands that will load the **ide-scsi** module and that provide the SCSI emulation, as well as govern the sequence in which SCSI (**sg** and **sr_mod**) an IDE CD (**ide-cd**) modules are loaded.

```
pre-install sg     modprobe ide-scsi # load ide-scsi before sg
pre-install sr_mod modprobe ide-scsi # load ide-scsi before sr_mod
pre-install ide-scsi modprobe ide-cd # load ide-cd   before ide-scsi
```

The following example shows the lines to add to the **/etc/modules.conf** file to have two IDE CD drives (hdc and hddd) emulate SCSI CD drives (scd0 and scd1). Notice the double quotes around the **ide-cd** options on the first line.

```
options ide-cd "ignore=hdc ignore=hdd"
alias scd0 sr_mod
alias scd1 sr_mod
pre-install sg modprobe ide-scsi
pre-install sr_mod modprobe ide-scsi
pre-install ide-scsi modprobe ide-cd
```

Once you have installed your SCSI emulation, you should check that your IDE drives are being recognized as SCSI drives. To do this, you run the **cdrecord** program with the **-scanbus** option. The following example shows two IDE CD drives now emulating SCSI CD drives. One is a Plextor IDE CD-RW drive (scd0) and the other is a Toshiba DVD-ROM drive (scd1).

```
# cdrecord -scanbus
Cdrecord 1.9 (i686-pc-linux-gnu) Copyright (C) 1995-2000 Jörg Schilling
Linux sg driver version: 3.1.17
Using libscg version 'schily-0.1'
scsibus0:
    0,0,0  0) 'PLEXTOR ' 'CD-R   PX-W1210A' '1.02' Removable CD-ROM
    0,1,0  1) 'TOSHIBA ' 'DVD-ROM SD-M1402' '1010' Removable CD-ROM
    0,2,0  2) *
    0,3,0  3) *
    0,4,0  4) *
    0,5,0  5) *
    0,6,0  6) *
    0,7,0  7) *
```

You then need to change the device links for any IDE CD drives already installed. Your Linux CD Writing software and your **/etc/fstab** entries are designed to reference /etc/cdrom links. Originally these were set up to link to your IDE device files, such as **/dev/hdc**. You now have to change them to reference the SCSI CD device files, like **/dev/scd0**. For example, if you have two IDE CD drives, then you will have two device links called **/etc/cdrom** and **/etc/cdrom1**. If both IDE CD drives are now emulating SCSI drives, you have to change both the cdrom and cdrom1 links. You can place these rm and ln commands in the /etc/rc.d/rc.local file to have them automatically implemented when your system starts up.

1. First, erase the IDE CD-ROM links in the **/dev** directory. These are currently pointing to the IDE CD-ROM devices like **hdc** or **hdb**.
2.
```
rm -f /dev/cdrom
rm -f /dev/cdrom1
```

3. Then create them again to point to the corresponding SCSI devices for these drives. Use the **ln** command with the **-s** option to create symbolic links. These devices begin with the prefix scd (SCSI CD) and are numbered from 0, beginning with **scd0**, **scd1**, and so on.
4.
```
ln -s /dev/scd0  /dev/cdrom
ln -s /dev/scd1 /dev/cdrom1
```

5. As the links are specified in the **/etc/fstab** file, you can now mount and access the drives just using their mount point.

```
mount /mnt/cdrom
```

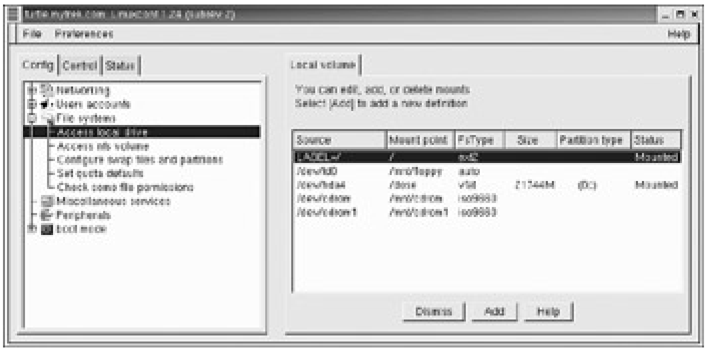
6. Finally, in the CD-ROM entry for your CD-R or CD-RW drive in the **/etc/fstab** file, you would specify the **rw** (read/write) option instead of the **ro** (read-only) option. In this example, **/dev/cdrom** links to **/dev/scd0**, which is a Plextor IDE CD-RW drive.

```
/dev/cdrom  /mnt/cdrom    iso9660 noauto,owner,kudzu,rw 0 0
```

Linuxconf Configuration for Local File Systems

Unless you are familiar with the **fstab** file, using Linuxconf to add and edit entries is easier-instead of editing entries directly. Linuxconf is available on Red Hat systems and can be installed on any major distribution. With Linuxconf, you can choose many of the configuration options using drop-down menus and check boxes. Once you finish making your entries, you can have Linuxconf generate a new **/etc/fstab** file incorporating your changes.

You can start Linuxconf from a window manager, a desktop, or a shell command line. The figures here show Linuxconf panels as they are displayed in Gnome. Other desktops and window managers show the same display. A Linuxconf shell command shows cursor-based lists and boxes. To access the Linuxconf file system configuration panels, you can select its entry in the main Linuxconf interface, or you can use the **fsconf** command to invoke a specialized window showing only file system options. Using the main Linuxconf interface, you select the Access local drive entry in the file systems list under the Config heading. This displays the information about the different file systems accessible on your local system. The source is the device name for the storage device. The name begins with **/dev**, the directory where device files are kept. For example, the name given to the first partition on the first hard drive is **hda1**. Its device filename is **/dev/hda1**. On most Linux distributions, a CD-DROM is given the device name **/dev/cdrom**. Names for additional CD-ROMS vary, depending on whether they are SCSI or IDE devices. On PCs, a second IDE CD-ROM could have the name **/dev/hdd**. The mountpoint and file system type are shown along with the size, partition type, and whether it is mounted. Figure 32-2 shows the Linuxconf Access local drive panel.

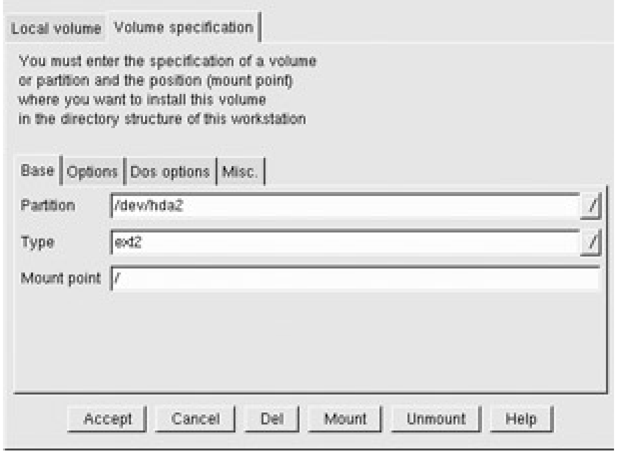


You can edit, add, or delete mounts
Select [Add] to add a new definition

Source	Mount point	FsType	Size	Partition type	Status
LABEL=/	/	ext2			Mounted
/dev/fd0	/mnt/floppy	auto			
/dev/hda4	/dos	vfat	21344M	(0C)	Mounted
/dev/cdrom	/mnt/cdrom	iso9660			
/dev/cdrom1	/mnt/cdrom1	iso9660			

Figure 32-2: Linuxconf file system configuration

To add a new entry, click the Add button. This displays panels for file system information and options. The Base tab prompts you to enter the file system's device name, its file system type, and its mountpoint, as shown in Figure 32-3. The box labeled Partition is where you enter the file system's device name. This box holds a drop-down menu listing the different hard disk partitions on your hard drive. If you are making an entry for one of those partitions, you can select it from there. If you are making an entry for a CD-ROM or floppy drive, you must enter the name yourself. The Type box is where you enter the file system type. The box contains a drop-down menu that lists the different file system types from which you can choose. Select the type from this menu. See Table 32-7 for a listing of file system types supported. The Mountpoint box is where you enter the mountpoint-the directory on your main file system where you attach this file system. For example, a floppy disk is usually attached to the **/mnt/floppy** directory. When you mount a floppy disk, you find its files in that directory. The directory can actually be any directory on your file system. If the directory does not exist, then Linuxconf asks to create it for you. Figure 32-3 shows an example of the Base tab for a Linux partition. The device name in this example is **/dev/hda2**, the file system type is **ext3** (standard for Linux), and the mountpoint is / (this is the root directory). If you were mounting a windows file system, you would use the **vfat** type, and for DOS partitions you would use **msdos**.



You must enter the specification of a volume or partition and the position (mount point) where you want to install this volume in the directory structure of this workstation

Partition /dev/hda2

Type ext2

Mount point /

Figure 32-3: Linuxconf Local Volume Base tab for adding or editing file systems

The Options tab lists different mount options in the form of check boxes. You can select options, such as whether you don't want the file system automatically mounted at boot, or if you want to let normal users mount it.

If your Linux system shares a hard drive with a Windows system, you may want to add mount entries for the Windows partition. This way, those partitions would be accessible by your Linux system. Linux can access any file on an Windows partition, though it cannot run its programs. For example, files could be downloaded by your Linux system and saved directly on a Windows partition. To create a mount entry for a Windows partition, you would enter its hard disk partition name in the Partition box, and enter **vfat** for its file system type.

When you finish, click the Accept button. This returns you to the Local volume panel and you then see the new entry displayed. Linuxconf does not actually implement these changes on your system until you click the Act/Changes entry in the Linuxconf File menu. When you do this, Linuxconf generates a new **/etc/fstab** file, replacing the previous one.

If you create an MS-DOS partition, you may want to control access to it. As a single- user system, MS-DOS does not implement any of the user access controls found on a multiuser system like Linux. For this reason, MS-DOS systems are mounted by default with global access, allowing any user on your Linux system full read and write access to the partition. If you want to control access, say to a particular user, you need to specify the controls you want on the MS-DOS Options panel. Here you will find entries for the user and the group, and the default permissions for that user and group. This way you can limit access to a single user or to users belonging to a particular group. DOS and Linux implement files differently, particularly in the way the end-of-file indicator is represented. When copying a Linux text file to DOS directory, a DOS end-of-file character should be inserted in it so that it can be read in DOS. Normally the translation setting is **auto**, which will translate text files but not binary ones, such as programs or archives (**text** will translate all files and **binary** will translate none).

You can change any of the current entries by simply double-clicking its entry. This displays the Volume specification panel for that entry. You can then change any of the options or the base configuration. For example, to use a different mountpoint, type in a new directory in the Mountpoint box. If you want to delete the entry, click the DEL button.

Webmin Configuration for Local File Systems

To use Webmin to administer your file system, you first select Disk and Network Filesystems on the System panel. This opens a page listing all the file systems listed in your **fstab** file. The first column lists their mountpoints, and these are links to pages where you can configure individual entries. To change any of the entries, you can click on their mountpoint directory to open an Edit Mount page. Here, you can set the mountpoint directory, the device mounted, and the different mountpoint options.

If you want to add a new file system, you first select the file system type from the pop-up menu at the bottom of the page. This menu will have as its default Linux Native Filesystem. Once you have selected the file system type, click on the Add Mount button next to it. This opens a page where you can enter the directory for the mountpoint, the device to be mounted, and the mount options.

Mounting File Systems Using Linuxconf, Webmin, KDE, and Gnome

Once you enter a new file system configuration and activate the changes to implement it on your system, you can then actually mount the file system. By default, file systems are mounted automatically-though certain file systems, such as CD-ROMs and floppy disks, are normally mounted manually.

Both the Gnome and K Desktop provide easy-to-use desktop icons for mounting and unmounting your file systems. Normally, these are used for manually mounted devices, such as CD-ROMs, tapes, and floppy disks. Hard disk partitions are usually mounted automatically at boot time. If a CD-ROM or floppy device can be mounted with the **user** option, then normal users on your system can also have CD-ROM and floppy icons for mounting and unmounting CD-ROMs and floppy disks.

On Gnome 1.2, an icon is automatically generated for your CD-ROM devices when you first insert a CD-ROM disk. On Gnome 1.4, you select the CD-ROM to mount from the Desktop menu Disks submenu (right-click anywhere on the desktop). Right-clicking the icon displays a menu from which you can select an entry to mount or unmount the CD-ROM. For a floppy disk, you right-click on the Floppy drive icon and select Mount to mount the floppy. Be sure to unmount it before you remove the floppy disk. You can also install a Mount applet in a Gnome panel that enables you to mount a floppy, CD-ROM, or partition by simply clicking its Panel icon (See Chapter 8 for more details).

On the K Desktop, KDE desktop files and their icons for your CD-ROM and floppy devices were created for you when KDE was installed. Should you want to add a new CD-ROM or floppy drive, right-click the desktop and select the entry either for the CD-ROM or floppy device from the Create New entry in the pop-up menu. Enter a name for the desktop file in the General panel. In the Device panel, enter the device name, mountpoint, and file system type. You can also select a mount and unmount icon.

To mount a file system manually with Linuxconf, select Control Configured Local Drives in the Mount/Unmount File Systems list in the Control Panel. This displays a listing of your local file systems. When you click an entry, you are then asked if you want to mount the file system. You can also use this panel to unmount a file system. If a file system is already mounted, clicking it displays a dialog box asking if you want to unmount the file system. Remember, when you mount a CD-ROM or floppy disk, you cannot then simply remove it to put in another one. You first have to unmount the CD or floppy disk before you can take it out and put in another one.

Using the Access local drive configuration panel to mount a file system is also possible. Click the Mount button in a file system's Volume Specification panel. An Unmount button also unmounts it. You can use this method when checking to see if a new system mounts correctly.

With Webmin, you first display the Disk and Network Filesystems page (accessible from the System panel). Click on the mount directory for the file system you want to mount, and then select Mount in the Edit Mount page and save the changes.

The mount and umount Commands

You can also mount or unmount any file system using the **mount** and **umount** commands. You enter these commands on a shell command line. In a window manager or desktop, you can open a terminal window and enter the command there, or you can simply use your login shell. The mount operations discussed in the previous sections use the **mount** command to mount a file system. Normally, mounting file systems can only be done as the root user (unless the device is user mountable). This is a system administration task and cannot be performed by a regular user. To mount a file system, be sure to log in as the root user. Table 32-9 lists the different options for the **mount** command.

Table 32-9: Themount Command

Mount Option	Description
-f	Fakes the mounting of a file system. You use it to check if a file system can be mounted.
-v	Verbose mode. Mount displays descriptions of the actions it is taking. Use with **-f** to check for any problems mounting a file system, **-fv**.
-w	Mount the file system with read and write permission.
-r	Mount the file system with only read permission.
-n	Mount the file system without placing an entry for it in the **mstab** file.
-t *type*	Specify the type of file system to be mounted. See Table 32-7 for valid file system types.
-a	Mount all file systems listed in **/etc/fstab**.
-o *option-list*	Mount the file system using a list of options. This is a comma-separated list of options following **-o**. See Table 32-8 for a list of the options and the Man pages for mount.

The **mount** command takes two arguments: the storage device through which Linux accesses the file system, and the directory in the file structure to which the new file system is attached.

The *mountpoint* is the directory on your main directory tree where you want the files on the storage device attached. The *device* is a special device file that connects your system to the hardware device. The syntax for the **mount** command is as follows:

```
# mount device mountpoint
```

Device files are located in the **/dev** directories and usually have abbreviated names ending with the number of the device. For example, **fd0** may reference the first floppy drive attached to your system. On Linux systems operating on PCs, the hard disk partitions have a prefix of **hd,** followed by an alphabetic character that labels the hard drive, and then a number for the partition. For example, **hda2** references the second partition on the first hard drive. In most cases, you can use the **man** command with a prefix to obtain more detailed information about that kind of device. For example, **man sd** displays the Man pages for SCSI devices. The following example mounts a floppy disk in the first floppy drive device (**fd0**) to the **/mydir** directory. The mountpoint directory needs to be empty. If you already have a file system mounted there, you will receive a message that another file system is already mounted there and that the directory is busy. If you mount a file system to a directory that already has files and subdirectories in it, those will be bypassed, giving you access only to the files in the mounted file system. Unmounting the file system restores access to the original directory files.

```
# mount /dev/fd0 /mydir
```

For any partition with an entry in the **/etc/fstab** file, you can mount it using only the mount directory specified in its **fstab** entry. You needn't enter the device filename. The **mount** command looks up the entry for it in the **fstab** file, using the directory to identify the entry and, in that way, find the device name. For example, to unmount the **/dev/hda1** DOS partition in the previous example, the **mount** command only needs to know the directory it is mounted to-in this case, **/mnt/dos**.

```
# mount /mnt/dos
```

If you want to replace one mounted file system with another, you must first explicitly unmount the one already mounted. Say you have mounted a floppy disk, and now you want to take it out and put in a new one. You must unmount that floppy disk before you can put in and mount the new one. You unmount a file system with the **umount** command. The **umount** command can take as its argument either a device name or the directory where it was mounted. Here is the syntax:

```
# umount device-or-mountpoint
```

The following example unmounts the floppy disk mounted to the **/mydir** directory:

```
# umount /dev/fd0
```

Using the example where the device was mounted on the **/mydir** directory, you could use that directory to unmount the file system:

```
# umount /mydir
```

One important constraint occurs on the **umount** command. You can never unmount a file system in which you are currently working. If you change to a directory within a file system that you then try to unmount, you receive an error message saying the file system is busy. For example, suppose you mount the Red Hat CD-ROM on the **/mnt/ cdrom** directory and then change to that **/mnt/cdrom** directory. If you decide to change CD-ROMs, you first have to unmount the current one with the **umount** command. This will fail because you are currently in the directory in which it is mounted. You first have to leave that directory before you can unmount the CD-ROM.

```
# mount /dev/hdc /mnt/cdrom
# cd /mnt/cdrom
# umount /mnt/cdrom
umount: /dev/hdd: device is busy
# cd /root
# umount /mnt/cdrom
```

If other users are using a file system you are trying to unmount, you can use the **lsof** or the **fuser** commands to find out who they are.

If you are unsure of the type of file system that the floppy disk holds, then you can mount it specifying the **auto** file system type with the **-t** option. Given the **auto** file system type, **mount** will attempt to automatically detect the type of file system on the floppy disk.

```
# mount -t auto /dev/fd0 /mydir
```

Mounting Floppy Disks

To access a file on a floppy disk, you first have to mount that disk onto your Linux system. The device name for your floppy drive is **fd0**, and it is located in the directory **/dev**. Entering **/dev/fd0** references your floppy drive. Notice the number **0** after **fd**. If you have additional floppy drives, they are represented by **fd1**, **fd2**, and so on. You can mount to any directory you want. Red Hat creates a convenient directory to use for floppy disks, **/mnt/floppy**. The following example mounts the floppy disk in your floppy drive to the **/mnt/floppy** directory:

```
# mount /dev/fd0 /mnt/floppy
```

Remember, you are mounting a particular floppy disk, not the floppy drive. You cannot simply remove the floppy disk and put in another one. The **mount** command has attached those files to your main directory tree, and your system expects to find those files on a floppy disk in your floppy drive. If you take out the disk and put another one in, you get an error message when you try to access it.

To change disks, you must first unmount the floppy disk already in your disk drive; then, after putting in the new disk, you must explicitly mount that new disk. To do this, use the **umount** command. Notice no *n* is in the **umount** command.

```
# umount /dev/fd0
```

For the **umount** operation, you can specify either the directory it is mounted on or the **/dev/fd0** device.

```
# umount /mnt/floppy
```

You can now remove the floppy disk, put in the new one, and then mount it.

```
# mount /mnt/floppy
```

When you shut down your system, any disk you have mounted is automatically unmounted. You do not have to unmount it explicitly.

Mounting CD-ROMs

You can also mount CD-ROM disks to your Linux system using the **mount** command. On Red Hat, the directory **/mnt/cdrom** has been reserved for CD-ROM file systems. You see an entry for this in the **/etc/fstab** file. With such an entry, to mount a CD-ROM, all you have to do is enter the command **mount** and the directory **/mnt/cdrom**. You needn't specify the device name. Once mounted, you can access the CD-ROM through the **/mnt/cdrom** directory.

```
# mount /mnt/cdrom
```

As with floppy disks, remember you are mounting a particular CD-ROM, not the CD-ROM drive. You cannot just remove the CD-ROM and put in a new one. The **mount** command has attached those files to your main directory tree, and your system expects to find them on a disc in your CD-ROM drive. To change discs, you must first unmount the CD-ROM already in your CD-ROM drive with the **umount** command. Your CD-ROM drive will not open until you issue this command. Then, after putting in the new disc, you must explicitly mount that new CD-ROM. You can then remove the CD-ROM and put in the new one. Then, issue a **mount** command to mount it.

```
# umount /mnt/cdrom
```

If you want to mount a CD-ROM to another directory, you have to include the device name in the **mount** command. The following example mounts the disc in your CD-ROM drive to the **/mydir** directory. The particular device name for the CD-ROM in this example is **/dev/hdc**.

```
# mount /dev/hdc /mydir
```

To change discs, you have to unmount the CD-ROM already in your CD-ROM drive, and then, after putting in the new disc, you must explicitly mount that new CD-ROM.

```
# umount /mydir
```

You can now remove the CD-ROM and put in the new one. Then, issue a **mount** command to mount it.

```
# mount /dev/hdc /mydir
```

Mounting Hard Drive Partitions: Linux and Windows

You can mount either Linux or Windows hard drive partitions with the **mount** command. However, it is much more practical to have them mounted automatically using the **/etc/ fstab** file as described in the next section. The Linux hard disk partitions you created during installation are already automatically mounted for you. To mount a Linux hard disk partition, enter the **mount** command with the device name of the partition and the directory to which you want to mount it. IDE hard drives use the prefix **hd**, and SCSI hard drives use the prefix **sd**. The next example mounts the Linux hard disk partition on **/dev/hda4** to the directory **/mnt/mydata**:

```
# mount -t ext3 /dev/hda4 /mnt/mydata
```

You can also mount a Windows partition and directly access the files on it. As with a Linux partition, you use the **mount** command, but you also have to specify the file system type as Windows. For that, use the **-t** option, and then type **vfat** (**msdos** for MS-DOS). In the next example, the user mounts the Windows hard disk partition **/dev/hda1** to the Linux file structure at directory **/mnt/windows**. The **/mnt/windows** directory is a common designation for Windows file systems, though you can mount it in any directory (/**mnt/dos** for MS-DOS). If you have several Windows partitions, you could create a Windows directory and then subdirectory for each drive using the drive's label for letter, such as **/mnt/Windows/a** or **/mnt/Windows/mystuff**. Be sure you have already created the directory.

```
# mount -t vfat /dev/hda1 /mnt/windows
```

Formatting File Systems: mkfs, mke2fs, mkswap, and fdisk

If you want to mount a new partition from either a new hard drive or your current drive, you must first create that partition using the Linux fdisk and format it with mkfs. Once it is created and formatted, you can then mount it on your system. If you need to create a swap partition, you use mkswap. To format Linux ext3 partitions, you can use mke2fs instead of mkfs. And for DOS partitions, you can use mkdosfs. mke2fs has its own set of options geared to technical aspects such as block and fragment sizes. To format standard Linux partitions, it is advisable to simply use mke2fs. The mkisofs tool will create a CD image. This is used primarily for creating CDs. Linux formatting and partition tools are listed in Table 32-10.

Table 32-10: Linux Partition and Formatting Tools

Tool	**Description**
fdisk	Create and delete partitions.
cfdisk	Screen-based interface for fdisk.
mkfs	Format a partition or floppy disk using specified filesystem type. Front end to format utilities.
mke2fs	Format an ext2 Linux partition.
mke2fs -j	Format an ext3 Linux partition.
mkswap	Format a swap partition.
mkdosfs	Format a DOS partition.
mkisofs	Create an ISO CD-ROM disk image.
kfloppy	KDE utility to format a floppy disk.

To start fdisk, enter **fdisk** on the command line. This brings up an interactive program you can use to create your Linux partition. Be careful using Linux fdisk. It can literally erase your entire hard disk if you are not careful. The Linux fdisk operates much as described in the installation process discussed in Chapter 2. The command **n** creates a new partition, and the command **t** enables you to set its type to that of a Linux type, 83. Table 32-11 lists the fdisk commands.

Table 32-11: The fdisk Commands

Command	Description
a	Sets and unsets the bootable flag for a partition.
c	Sets and unsets the DOS compatibility flag.
d	Deletes a partition.
l	List partition types.
m	Displays a listing of fdisk commands.
n	Creates a new partition.
p	Prints the partition table, listing all the partitions on your disk.
q	Quits without saving changes. Use this to abort an fdisk session if you made a mistake.
y	Select the file system type for a partition.
v	Verify the partition table.
w	Write partition table to disk and exit. At this point, the changes are made irrevocably.
x	Display a listing of advanced fdisk commands. With these, you can set the number of cylinders, sectors, and heads; print raw data; and change the location of data in the partition table.

Hard disk partitions are named with **hd** (IDE drive) or **sd** (SCSI drives), followed by an alphabetic letter indicating the hard drive, and then a number for the partition on the hard drive. They can belong to any operating system, such as MS-DOS, OS/2, or Windows NT, as well as Linux. The first partition created is called **hda1**-the first partition on the first IDE hard drive, *a*. If you add another partition, it will have the name **hda2**. If you add a new IDE hard drive, its first partition will have the name **hdb1**.

Once you create your partition, you have to format it. For this, use the **mkfs** command and the name of the hard disk partition. A hard disk partition is a device with its own device name in the **/dev** directory. You must specify its full pathname with the **mkfs** command. For example, the second partition on the first hard drive has the device name **/dev/hda4**. You can now mount your new hard disk partition, attaching it to your file structure. The next example formats that partition:

```
# mkfs -t ext3 /dev/hda4
```

mkfs is a front end that calls other tools to perform the actual formatting operation. For example, to format a Linux partition, mkfs uses mke2fs -j. For a windows or DOS partition, it uses mkdosfs. To create an ext2 Linux partition, you could just as easily use mke2fs with the -j option, and not have to specify a type, as shown here (without the -j option it creates an ext2 file system)

```
# mke2fs -j /dev/hda4
```

To format a floppy disk, use the **mkfs** command. This creates a Linux file system on that disk. Be sure to specify the ext3 file system type with the **-t ext3** option (see Table 32-12). Once it is formatted, you can then mount that file system. The **mkfs** command takes as its arguments the device name and the number of memory blocks on the disk (see Table 32-12). At 1,000 bytes per block, 1,400 formats a 1.44MB disk. You do not first mount the blank disk; you simply put it in your floppy drive and enter the **mkfs** command with its arguments. The next example formats a 1.44MB floppy disk:

```
# mkfs -t ext3 /dev/fd0 1400
```

With **mke2fs**, you could use:

```
# mke2fs -j /dev/fd0 1400
```

Table 32-12: The mkfs Options

Option	**Description**
Blocks	Number of blocks for the file system. There are 1,440 blocks for a 1.44MB floppy disk.
-t *file-system-type*	Specify the type of file system to format. The default is the standard Linux file system type, ext3.
fs -*options*	Options for the type of file system specified.
-V	Verbose mode. Displays description of each action **mkfs** takes.
-v	Instructs the file system builder program that **mkfs** invokes to show actions it takes.
-c	Checks a partition for bad blocks before formatting it (may take some time).
-l *file-name*	Reads a list of bad blocks.

If you have the K Desktop installed, you can use the kfloppy utility to format your floppy disks. kfloppy enables you to choose an MS-DOS or Linux file system type. For MS-DOS disks, you can choose a quick or full format.

If you want to create a swap partition, you first use fdisk to create the partition if it does not already exist, and then you use the **mkswap** command to format it as a swap partition. **mkswap** will format the entire partition unless otherwise instructed. It takes as its argument the device name for the swap partition.

```
mkswap /dev/hda5
```

Note During installation, Red Hat will recognize the IDE CD-R and CD-RW drives you have installed on your system, and will include the ide-scsi kernel parameter automatically as part of either your LILO or Grub boot loader configurations. You will not need to perform any of the specific configuration tasks described in this section.

Configuring RAID devices

Redundant array of independent devices (RAID) is method of storing data across several disks to provide greater efficiency and redundancy. In effect, you can have several hard disks treated as just one hard disk by your operating system. RAID then efficiently stores and retrieves data across all these disks, instead of having the operating system separately access each one as a separate file system. Lower-level details of storage and retrieval are removed from concern of the operating system. This allows greater flexibility in adding or removing hard disks, as well as implementing redundancy in the storage system to provide greater reliability. With RAID, you can have several hard disks that are treated as one virtual disk, where some of the disks are used as real-time mirrors, duplicating data.

RAID can be implemented on a hardware or software level. On a hardware level, you can have hard disks connected to a RAID hardware controller, usually a special PC card. Your operating system then accesses storage through the RAID hardware controller. Alternatively, you can implement RAID as a software controller, letting a software RAID controller program manage access to hard disks treated as RAID devices. The software version lets you use IDE hard disks as RAID disks. Linux uses the MD driver, supported in the 2.4 kernel, to implement a software RAID controller.

Note Before you can use RAID on your system, make sure it is implemented on your kernel. If not, you will have to reconfigure and install a new version of the kernel (see Chapter 34). Check the Multi-Driver Support component in your kernel configuration. You can specify support of any or all of the RAID levels.

RAID can be implemented at different levels depending on whether you want efficiency, redundancy, or reconstruction capability. For efficiency, RAID stores data using disk stripping, where data is organized into standardized strips that can be stored across the RAID drives for faster access (level 0). Redundancy is implemented with mirroring. With mirroring, the same data is written to each RAID drive (level 1). Each disk has a complete copy of all the data written, so that if one or more fails, the others still have your data. Redundancy can be very inefficient and take up a great deal of storage. It is usually implemented on RAID arrays of only two disk drives, where one is used as a real time backup. As an alternative, data can be reconstructed using parity information in case of a hard drive crash. Parity information is saved instead of full duplication of the data (level 5). Parity information takes up the space equivalent of one drive, leaving most of the space on the RAID drives free for storage. On Red Hat, RAID supports three levels as well as a simple linear implementation (see Table 32-13).

Table 32-13: RAID Levels Supported on Red Hat

RAID Level	**Description**
0	Implements disk stripping across drives with no redundancy.
1	Implements a high level of redundancy. Each drive is treated as a mirror for all data.
5	Implements a data reconstruction capability using parity information distributed across all drives. The parity information takes up the equivalent of one drive.
linear	Simply treats RAID hard drives as one virtual drive with no stripping, mirroring, or parity reconstruction.

Note You can create and format RAID drives on Red Hat during installation. At that time you can use either fdisk or Disk Druid to create your RAID partitions and devices. Consult the Red Hat Customization Guide for details. Select the custom configuration for creating your partitions. You can then use Disk Druid (recommended) or fdisk to create RAID partitions.

This section will discuss Linux software RAID devices as they are implemented using Red Hat RAID tools (see Table 32-14). A RAID device is called an **md** device because it uses the MD driver. These devices are already defined on your Red Hat Linux system in the **/etc/dev** directory, starting with **md0**. **/dev/md0** is the first RAID device, and **/dev/md1** is the second, and so on. Each RAID device, in turn, will use hard disk partitions, where each partition contains an entire hard disk. These partitions are usually referred to as RAID disks, whereas a RAID device is an array of the RAID disks its uses.

Table 32-14: Red Hat RAID Tools

Tool	Description
mkraid	Creates (configures) a RAID devices from a set of block devices, initializing them
raidstart	Activates RAID devices
raid0start	Activates older non-persistent linear and RAID 0 RAID devices
raidstop	Turns off a RAID device, un-configuring it

Note The term *device* can be confusing as it is also used to refer to the particular hard disk partitions, also devices, that make up a RAID device. In fact, a RAID device is an array of hard disk partitions, with each partition taking up an entire hard disk. You can think of a RAID device as consisting of a set (array) of hard disks (devices). So you have the RAID device, which is an array of hard disk devices.

If you created your RAID devices and their partitions during the Red Hat installation process, you should already have working RAID devices. Your RAID devices will be configured in the **/etc/raidtab** file, and the status of your RAID devices will be listed in the **/proc/mdstat** file. You can manually start or stop your RAID devices with the **ra**idstart and **raidstop** commands. The **-a** option will operate on all of them, although you can specify particular devices if you want.

Creating and installing a new RAID device involves the following steps:

- Make sure that your kernel supports the kind of RAID device you are creating.
- If you have not already done so, create the RAID disks (partitions) you will use for your RAID device.
- Configure your RAID device (**/dev/md***n*) in the **/etc/raidtab** file, specifying the RAID disks to use.
- Create your RAID device with **mkraid**.
- Activate the RAID device with **raidstart**.
- Create a file system on the RAID device (**mke2fs**), and then mount it.

To add new RAID devices or to create them in the first place, you will need to manually create the hard disk partitions they will use and then configure RAID devices to use those partitions. To create a hard disk partition for use in a RAID array, you use fdisk and specify **fd** as the file system type. You invoke fdisk with the device name of the hard disk you want to create the partition on. Be sure to specify **fd** as the partition type. The following example will invoke fdisk for the hard disk **/dev/hdb** (the second hard disk on the primary IDE connection).

```
fdisk /dev/hdb
```

Although technically partitions, these hard disk devices are referred to as disks in RAID configuration documentation and files, so that is how they will be referred to from this point on.

Once you have your disks, you then need to configure RAID devices to use them. RAID devices are configured in the **/etc/raidtab** file. Here you create a **raiddev** entry for each RAID device and specify which disks they will use. **readdev** specifies the name of the RAID device you are configuring, such as **/dev/md0** for the first RAID device.

```
raiddev  /dev/md0
```

You then specify the level for the RAID device such as 0, 4, or 5 (**raidlevel**). Next, you add any options you may need, along with the list of disks making up the RAID array. A disk is defined with the **device** entry, and its position in the RAID array with the **raid-disk** option. The configuration directives and options are listed in Table 32-15. A sample entry for the **/etc/raidtab** file is shown here:

```
        raiddev /dev/md0
            raid-level              5
            nr-raid-disks           3
            nr-spare-disks          1
            persistent-superblock   1
            chunk-size              4
            parity-algorithm        left-symmetric

            device                  /dev/hdb1
            raid-disk               0
            device                  /dev/hdc1
            raid-disk               1
            device                  /dev/hdd1
            raid-disk               0
```

Table 32-15: raidtab Options

Directives and Options	Description
raiddev *device*	Starts a configuration section for a particular RAID device.
raid-level *num*	The RAID level for the RAID device, such as 0, 1, 4, 5, and -1 (linear).
device *disk-device*	The disk device (partition) to be added to the RAID array. The number of device entries specified for a RAID device must match that specified by **nr-raid-disk**.
nr-raid-disks *count*	Number of raid devices in an array. Each RAID device section must have this directive. Maximum limit is 12 (256 experimental).
nr-spare-disks *count*	Number of spare devices in the array. Used only for RAID 4 and RAID 5. Kernel must be configured to and allow the automatic addition of new RAID disks as needed. Can add and remove spare disks with **raidhotadd** and **raidhotremove**.
persistent-superblock *0/1*	Specifies whether newly created RAID arrays should use a persistent superblock. Used to help the kernel safely detect

Table 32-15: raidtab Options

Directives and Options	Description
	the RAID array. RAID array information is kept in a superblock on each RAID member.
chunk-size *size*	Sets the stripe size to size bytes, in powers of 2.
device *devpath*	Adds the most recently devined device to the list of devices that make up the RAID system.
raid-disk *index*	Inserts the most recently devined RAID device at the specified position in the RAID array.
spare-disk *index*	Inserts the most recently devined RAID device as a spare device at the specified position in the RAID array.
parity-disk *index*	The most recently devined device is used as the parity device, placing it at the end of the RAID array.
parity-algorithm *algoritm*	For RAID 5 devices, specifies the parity algorithm to use: left-asymmetric, right-asymmetric, left-symmetric, or right-symmetric.
failed-disk *index*	The most recently defined device is added to a RAID array as a failed device at the specified position.

The previous example configures the RAID device **/dev/md0** as a RAID 5 (**raid-level 5**) device. There are three disks (partitions) that make up this RAID array, **/dev/hdb1**, **/dev/hdc1**, and **/dev/hdd1**, of which /**dev/hdb1** is the first and **/dev/hdc1** is the second. There is one spare disk, **/dev/hdd1**. There are three RAID disks altogether (**nr-raid-disks**) and one spare partition (**nr-spare-disks**). The RAID file system uses persistent super blocks (**persistent-superblock**) to hold file system configuration information. The **parity-algorithm** option is used for RAID 5 devices to specify the type of parity algorithm to use for parity restoration-in this example, left-symmetric. Red Hat also provides a set of RAID tools for creating and maintaining RAID partitions and devices.

Once you have configured your RAID devices in the **/etc/raidtab** file, you then use the **mkraid** command to create your RAID devices. **mkraid** takes as its argument the name of the RAID device, such as **/dev/md0** for the first RAID device. It then locates its entry in the **/etc/raidtab** file and uses that configuration information to create the RAID file system on that device. You can specify an alternative configuration file with the **-c** options, if you wish. **mkraid** operates as a kind of **mkfs** command for the RAID device, initializing the partitions and creating the RAID file systems. Any data on the partitions making up the RAID array will be erased.

```
mkraid /dev/md0
```

Once you have created your RAID devices, you can then activate them with the **raidstart** command. **raidstart** makes your RAID file system accessible. **raidstart** takes as its argument the name of the RAID device you want to start. The **-a** option will activate all RAID devices.

```
raidstart /dev/md0
```

Once they are activated, you can then create file systems on the RAID devices and mount those file systems. The following example creates a standard Linux file system on the **/dev/md0** device.

```
mke2fs /dev/md0
```

The user then creates a directory called **/myraid** and mounts the RAID device there.

```
mkdir /myraid
mount /dev/md0 /myraid
```

Should you plan to use your RAID device for maintaining your user directories and files, you would mount the RAID device as your **/home** partition. Such a mount point might normally be used if you created your RAID devices when you installed your system. To transfer your current home directories to a RAID device, first back them up on another partition, and then mount your RAID device, copying your home directories to it.

If you decide to change your RAID configuration or add new devices, you first have to deactivate your currently active RAID devices. To deactivate a RAID device you use the **raidstop** command. Be sure to close any open files and unmount any file systems on the device first.

```
umount /dev/md0
raidstop /dev/md0
```

The **raidhotadd** and **raidhotremove** commands are used to add and remove partitions from an active RAID array. You use **raidhotadd** to add a spare partition and **raidhotremove** to remove any partitions that have failed.

Note **raidstop**, **raidhotadd**, and **raidhotremove** are simply links to the **raidstart** command. They run the **raidstart** command with certain options.

Backup and Restore File Systems: dump and restore

You can back up and restore your system with the dump and restore utilities. dump can back up your entire system or perform incremental backups, saving only those files that have changed since the last backup. dump supports several options for managing the backup operation, such as specifying the size and length of storage media (see Table 32-16).

Table 32-16: dump Options

Option	**Description**
-0-9	Specify the dump level. A dump level 0 is a full backup, copying the entire file system (see also the **-h** option). dump level numbers above 0 perform incremental backups, copying all new or modified files new in the file system since the last backup at a lower level. The default level is 9.
-B *records*	Lets you specify the number of blocks in a volume, overriding the end-of-media detection or length and density calculations that dump normally uses for multivolume dumps.

Table 32-16: dump Options

Option	Description
-a	Lets dump bypass any tape length calculations and write until an end-of-media indication is detected. Recommended for most modern tape drives, and is the default.
-b *blocksize*	Lets you specify the number of kilobytes per dump record. With this option, you can create larger blocks, speeding up backups.
-d *density*	Specify the density for a tape in bits per inch (default is 1,600 BPI).
-h *level*	Files that are tagged with a user's "nodump"' flag will not be backed up at or above this specified level. The default is 1, which will not back up the tagged files in incremental backups.
-f *file/device*	Backs up the file system to the specified file or device. This can be a file or tape drive. You can specify multiple filenames, separated by commas. A remote device or file can be referenced with a preceding hostname, *hostname:file*.
-k	Use Kerberos authentication to talk to remote tape servers.
-M	Implements a multivolume backup, where the *file* written to is treated as a prefix and the suffix consisting of a numbered sequence from 001 is used for each succeeding file, *file*001, *file* 002, etc. Useful when backup files need to be greater than the Linux ext3 2GB file size limit.
-n	Notify operators should a backup need operator attention.
-s *feet*	Specify the length of a tape in feet. dump will prompt for a new tape when the length is reached.
-S	Estimate the amount of space needed to perform a backup.
-u	Write an entry for a successful update in the **/etc/dumpdates** file.
-W	Detects and displays the file systems that need to be backed up. This information is taken from the **/etc/dumpdates** and **/etc/fstab** files.
-w	Detects and displays the file systems that need to be backed up based only on information in **/etc/fstab**.

The dump utility uses *dump levels* to determine to what degree you want your system backed up. A dump level of 0 will copy file systems in their entirety. The remaining dump levels perform incremental backups, only backing up files and directories that have been created or modified since the last lower-level backup. A dump level of 1 will only back up files that have changed since the last level 0 backup. The dump level 2, in turn, will only back up files that have changed since the last level 1 backup (or 0 if there is no level 1), and so on up to dump level 9. You could run an initial complete backup at dump level 0 to back up your entire system, and then run incremental backups at certain later dates, having to back up only the changes since the full backup.

Using dump levels, you can devise certain strategies for backup of a file system. It is important to keep in mind that an incremental backup is run on changes from the last lower-level backup. For example, if the last backup was 6 and the next backup was 8, then the level 8 would back up everything from the level 6 backup. The sequence of the backups is important. If there were three backups with levels 3, then 6, and then 5, the level 5 backup would take everything from the level 3 backup, not stopping at level 6. Level 3 is the next-*lower*-level backup for level 5, in this case. This can make for some complex incremental backup strategies. For example, If you want each succeeding incremental backup to include all the changes from the preceding incremental backups, you could run the backups in descending dump level order. Given a backup sequence of 7, 6, and 5, with 0 as the initial full backup, then 6 would include all the changes to 7, because its next lower level is 0. Then, 5 would include all the changes for 7 and 6, also because its next lower level is 0, making all the changes since the level 0 full backup. A simpler way to implement this is to make the incremental levels all the same. Given an initial level of 0, and then two backups both with level 1, the last level 1 would include all the changes from the backup with level 0, since level 0 is the next *lower* level-not the previous level 1 backup.

Backups are recorded in the **/etc/dumpdates** file. This file will list all the previous backups specifying the file system they were performed on, the dates they were performed, and the dump level used. You can use this information to restore files from a specified backup. Recall that the **/etc/fstab** file records the dump level as well as the recommended backup frequency for each file system. With the **-W** option, dump will analyze both the **/etc/dumpdates** and **/etc/fstab** files to determine which file systems need to be backed up. The **dump** command with **-w** option just uses **/etc/fstab** to report the file systems ready for backup.

The **dump** command takes as its arguments the dump level, the device it is storing the backup on, and the device name of the file system that is being backed up. If the storage medium (such as a tape) is too small to accommodate the backup, dump will pause and let you insert another. dump supports backups on multiple volumes. The **u** options will record the backup in the **/etc/dumpdates** file. In the following example, an entire backup (dump level 0) is performed on the file system on the **/dev/hda3** hard disk partition. The backup is stored on a tape device, **/dev/tape**.

```
dump -0u -f /dev/tape /dev/hda5
```

Note You can use the **mt** command to control your tape device. **mt** has options to rewind, erase, and position the tape. The **rmt** command controls a remote tape device.

The storage device can be another hard disk partition, but it is usually a tape device. When you installed your system, Red Hat most likely detected the device and set up **/dev/tape** as a link to it (just as it did with your CD-ROMs). If the link was not set up, you would have to create it yourself or use the device name directly. Tape devices can have different device names, depending on the model or interface. SCSI tape devices are labeled with the prefix **st**, with a number attached for the particular device. **st0** is the first SCSI tape device. To use it in the **dump** command, just specify its name.

```
dump -0u -f /dev/st0 /dev/hda5
```

Should you need to back up to a device located on another system on your network, you would have to specify that hostname for the system and the name of its device. The hostname is entered before the device name and delimited with a colon. In the following example, the user backs up file system **/dev/hda5** to the SCSI tape device with the name **/dev/st1** on the **rabbit.mytrek.com** system:

```
dump -0u -f rabbit.mytrek.com:/dev/st0 /dev/hda5
```

The **dump** command works on one file system at a time. If your system has more than one file system, you will need to issue a separate **dump** command for each.

Note You can use the system cron utility to schedule backups using dump at specified times.

You use the **restore** command to either restore an entire file system or to just retrieve particular files. restore will extract files or directories from a backup archive and copy them to the current working directory. Make sure you are in the directory you want the files restored to when you run restore. restore will also generate any subdirectories as needed. restore has several options for managing the restore operation (see Table 32-17).

Table 32-17: restore Options

Option	Description
-C	Lets you check a backup by comparing files on a file system with those in a backup.
-I	The interactive mode for restoring particular files and directories in a backup. A shell interface is generated where the user can use commands to specify files and directories to restore (see Table 32-18).
-R	Will instruct restore to request a tape that is part of a multivolume backup, from which to continue the restore operation. Helpful when multivolume restore operations are interrupted.
-r	Restore a file system. Make sure that a newly formatted partition has been mounted and that you have changed to its top directory.
-t	List the contents of a backup, or specified files in it.
-x	Will extract specified files or directories from a backup. A directory is restored with all its subdirectories. If no file or directory is specified, then the entire file system is restored.
-f *file/device*	Restores the backup on the specified file or device. Specify a hostname for remote devices.
-k	Use Kerberos authentication for remote devices.
-h	Extracts only the specified directories, without their subdirectories.
-M	Restores from multivolume backups where the *file* is treated as a prefix and the suffix is a numbered sequence, *file*001, *file*002.
-N	Displays the names of files and directories, does not extract them.
-T *directory*	Specifies a directory to use for the storage of temporary files. The default value is **/tmp**.
-v	The verbose mode, where each file and its file type that restore operates on is displayed.
-y	By default, restore will query the operator to continue if an error occurs, such as bad blocks. This option suppresses that query, allowing restore to automatically continue.

To recover individual files and directories, you run restore in an interactive mode using the (**-i**) option. This will generate a shell with all the directories and files on the tape, letting you select the ones you want to restore. When you are finished, restore will then retrieve from a backup only those files you selected. This shell has its own set of commands that you can use to select and extract files and directories (see Table 32-18). The following command will generate an interactive interface listing all the directories and files backed up on the tape in the **/dev/tape** device:

```
restore -ivf /dev/tape
```

Table 32-18: Restore Interactive Mode Shell Commands

Command	**Description**
add [*arg*]	Add files or directories to the list of files to be extracted. Such tagged files display a * before their names when listed with **ls**. All subdirectories of a tagged directory are also extracted.
cd *arg*	Changes the current working directory.
delete [*arg*]	Deletes a file or directory from the extraction list. All subdirectories for deleted directories will also be removed.
extract	Extracts files and directories on the extraction list.
help	Display a list of available commands.
ls [*arg*]	Lists the contents of the current working directory or a specified directory.
pwd	Displays the full pathname of the current working directory.
quit	Exits the restore interactive mode shell. The **quit** command does not perform any extraction, even if the extraction list still has items in it.
setmodes	Sets the owner, modes, and times for all files and directories in the extraction list. Used to clean up an interrupted restore.
verbose	In the verbose mode, each file is listed as it is extracted. Also, the **ls** command lists the inode numbers for files and directories.

This command will generate a shell encompassing the entire directory structure of the backup. You are given a shell prompt and you can use the **cd** command to move to different directories and the **ls** command to list files and subdirectories. You use the **add** command to tag a file or directory for extraction. Should you later decide not to extract it, you can use the **delete** command to remove from the tagged list. Once you have selected all the items you want, you enter the **extract** command to retrieve them from the backup archive. To quit the restore shell, you enter **quit**. The **help** command will list the restore shell commands.

Should you need to restore an entire file system, you would use restore with the **-r** option. You can restore the file system to any blank formatted hard disk partition of adequate size, including the file system's original partition. If may be advisable, if possible, to restore the file system on another partition and check the results.

Restoring an entire file system involves setting up a formatted partition, mounting it to your system, and then changing to its top directory to run the **restore** command. First you should use mkfs to format the partition to where you are restoring the file system, and then mount it onto your system. Then you use restore with the **-r** option and the **-f** option to specify the device holding the file system's backup. In the next example, the user formats the **/dev/hda5** partition, and then restores on that partition, the file system backup, currently on a tape in the **/dev/tape** device.

```
mkfs /dev/hda5
mount /dev/hda5 /mystuff
cd /mystuff
restore -rf /dev/tape
```

To restore from a backup device located on another system on your network, you would have to specify that hostname for the system and the name of its device. The hostname is entered before the device name and delimited with a colon. In the following example, the user restores a file system from the backup on the tape device with the name **/dev/tape** on the **rabbit.mytrek.com** system:

```
restore -rf rabbit.mytrek.com:/dev/tape
```

The mtools Utilities: msdos

Your Linux system provides a set of utilities, known as *mtools,* that enable you to access a floppy disk formatted for MS-DOS easily (see Table 32-19). The **mcopy** command enables you to copy files to and from an MS-DOS floppy disk in your floppy drive. No special operations, such as mounting, are required. With mtools, you needn't mount an MS-DOS partition to access it. For an MS-DOS floppy disk, place the disk in your floppy drive, and you can then use mtool commands to access those files. For example, to copy a file from an MS-DOS floppy disk to your Linux system, use the **mcopy** command. You specify the MS-DOS disk with **a:** for the A drive. Unlike normal DOS pathnames, pathnames used with mtool commands use forward slashes instead of backslashes. The directory **docs** on the A drive would be referenced by the pathname **a:/docs**, not **a:\docs**. Unlike MS-DOS, which defaults the second argument to the current directory, you always need to supply the second argument for **mcopy**. The next example copies the file **mydata** to the MS-DOS disk, and then copies the **preface** file from the disk to the current Linux directory. Notice that, unlike DOS, mtools uses forward slashes instead of backward slashes.

```
$ mcopy mydata a:
$ mcopy a:/preface.
```

Table 32-19: The mtools Access Commands

Command	Execution
mcopy *filename filename*	Copies a file to and from an MS-DOS disk or your Linux system. The following copies a file from an MS-DOS floppy disk to your Linux system:**mcopy a:**/*filename directory-or-filename* The following copies a file from Linux for an MS-DOS floppy disk in your floppy drive:**mcopy** *filename* **a:**/*filename*.
mcd *directory-name*	Changes directory on your MS-DOS file system.
mdir	Lists the files on an MS-DOS disk in your floppy drive.
mattrib	Change the attribute of an MS-DOS file.
mdel *filename*	Delete an MS-DOS file.
mformat	Adds an MS-DOS file system to a floppy disk.
mlabel	Makes a volume label.
mmd *directory-name*	Makes an MS-DOS directory.
mrd *directory-name*	Removes an MS-DOS directory.

Table 32-19: The mtools Access Commands	
Command	**Execution**
mread *filename filename*	Low-level read (copy) of an MS-DOS file to Unix.
mren *filename filename*	Renames an MS-DOS file.
mtype *filename*	Displays contents of an MS-DOS file.
mwrite *filename filename*	Low-level write (copy) a Unix file to MS-DOS.

You can use the **mdir** command to list files on your MS-DOS disk, and you can use the **mcd** command to change directories on it. The next example lists the files on the MS-DOS disk in your floppy drive, and then changes to the **docs** directory on that drive:

```
$ mdir a:
$ mcd a:/docs
```

Most of the standard MS-DOS commands are available as mtool operations. You can create MS-DOS directories with **mmd** and erase MS-DOS files with **mdel**. A list of mtool commands is provided in Table 32-19. For example, to display a file on drive **b:** on an MS-DOS 5 1/4-inch floppy drive, use **mtype** and the name of the file preceded by **b:/**.

```
$ mtype b:/readme
```

Access to MS-DOS partitions is configured by the **/etc/mtools.conf** file. This file lists several different default MS-DOS partitions and disk drives. Each drive or partition is identified with a particular device name. Entries for your floppy drives are already entered, using the device names **/dev/fd0** and **/dev/fd1** for the first and second floppy drives. An entry in the **/etc/mtools.conf** file takes the form of the drive label followed by the term "file" and the equal sign, and then the device name of the drive or partition you want identified with this label. The device name is encased in quotes. For example, assuming the first hard disk partition is an MS-DOS partition and has the device name of **/dev/hda1**, the following entry would identify this as the **c:** drive on an MS-DOS system:

```
drive c: file="/dev/hda1"
```

You must have the correct device name for your partition. These device names are listed in the **/etc/fstab** file and can also be viewed with the Linuxconf local drive access panel on your root user desktop. If you have an SCSI hard disk, the hard disk partition has the form of **sd,** followed by a character for the hard drive and a number for the partition in it. For example, **sda1** refers to the first partition on the SCSI hard drive. IDE hard drives have the form of **hd**, also followed by a character and a partition number-**hda1** refers to the first partition on an IDE hard drive.

On most distributions, a default **/etc/mtools.conf** file is installed for you (see the following **/etc/mtools.conf** example). This file has commented entries for the **c:** drive: one for an SCSI hard disk partition and one for an IDE partition. Both are commented out with a preceding #. If you have an IDE hard drive (as most users do), you need to remove the preceding # symbol from the entry for the IDE hard disks partition and leave the preceding # symbol in front of the entry for the SCSI partition. Also, if your MS-DOS partition on your IDE hard drive is not the first partition, you must change the device name. For example, if the MS-DOS partition is the second partition, the device name will be **/dev/hda2**. If you have several MS-DOS partitions, you can add entries for each one, assigning a different label to each. The following example assigns the **d:** label to the fourth hard disk partition on an IDE drive:

```
drive d: file="/dev/hda4"
```

/etc/mtools.conf

```
# Linux floppy drives
drive a: file="/dev/fd0" exclusive 1.44m
drive b: file="/dev/fd1" exclusive 1.44m
# First SCSI hard disk partition
#drive c: file="/dev/sda1"
# First IDE hard disk partition
drive c: file="/dev/hda1"
drive d: file="/dev/hda5"
#dosemu floppy image
drive m: file="/var/lib/dosemu/diskimage"
#dosemu hdimage
drive n: file="/var/lib/dosemu/diskimage" offset=3840
#Atari ramdisk image
drive o: file="/tmp/atari_rd" offset=136
mtools_lower_case=1
```

Once the DOS hard disk partitions are referenced, you can then use their drive letters to copy files to and from them to your Linux partitions. The following command copies the file **mydoc.html** to the **c:** partition in the directory **webstuff** and renames it **mydoc.htm**. Notice the use of forward slashes instead of backward slashes.

```
$ mcopy mypage.html c:/webstuff/mypag.htm
```

Because of the differences in the way DOS and Linux handle newlines in text files, you should use the **-t** option whenever copying a DOS text file to a Linux partition. The following command copies the **mydoc.txt** file from the **c:/project** directory to the **/newdocs** directory:

```
$ mcopy -t c:/project/mydoc.txt /newdocs
```

Archive Files and Devices: tar

The tar utility creates archives for files and directories. With tar, you can archive specific files, update them in the archive, and add new files as you want to that archive. You can even archive entire directories with all their files and subdirectories, all of which can be restored from the archive. The tar utility was originally designed to create archives on tapes. The term "tar" stands for tape archive. You can create archives on any device, such as a floppy disk, or you can create an archive file to hold the archive. The tar utility is ideal for making backups of your files or combining several files into a single file for transmission across a network.

Note As an alternative to tar you can use pax. pax is designed to work with different kinds of Unix archive formats such as cpio, bcpio, and tar. You can extract, list, and create archives. pax is helpful if you are handling archives created on Unix systems that are using different archive formats.

On Linux, tar is often used to create archives on devices or files. You can direct tar to archive files to a specific device or a file by using the **f** option with the name of the device or file. The syntax for the **tar** command using the **f** option is shown in the next example. The device or filename is often referred to as the archive name. When creating a file for a tar archive, the filename is usually given the extension **.tar**. This is a convention only, and is not required. You can list as many filenames as you want. If a directory name is specified, then all its subdirectories are included in the archive.

```
$ tar optionsf archive-name.tar directory-and-file-names
```

To create an archive, use the **c** option. Combined with the **f** option, **c** creates an archive on a file or device. You enter this option before and right next to the **f** option. Notice no preceding dash is before a tar option. Table 32-20 lists the different options you can use with tar. In the next example, the directory **mydir** and all its subdirectories are saved in the file **myarch.tar**. In this example, the **mydir** directory holds two files, **mymeeting** and **party**, as well as a directory called **reports** that has three files: **weather**, **monday**, and **friday**.

Table 32-20: File Backups: tar

Command	**Execution**
tar *options files*	Backs up files to tape, device, or archive file.
tar *options*f *archive_name filelist*	Backs up files to a specific file or device specified as *archive_name. filelist;* can be filenames or directories.
Option	
c	Creates a new archive.
t	Lists the names of files in an archive.
r	Appends files to an archive.
U	Updates an archive with new and changed files; adds only those files modified since they were archived or files not already present in the archive.
w	Waits for a confirmation from the user before archiving each file; enables you to update an archive selectively.
x	Extracts files from an archive.
m	When extracting a file from an archive, no new timestamp is assigned.
M	Creates a multiple-volume archive that may be stored on several floppy disks.
f *archive-name*	Saves the tape archive to the file archive name, instead of to the default tape device; when given an archive name, the **f** option saves the tar archive in a file of that name.
f *device-name*	Saves a tar archive to a device such as a floppy disk or tape. **/dev/fd0** is the device name for your floppy disk; the default device is held in **/etc/default/tar-file**.
v	Displays each filename as it is archived.
z	Compresses or decompresses archived files using gzip.

Table 32-20: File Backups: tar	
Command	**Execution**
j	Compresses or decompresses archived files using bzip2.

```
$ tar cvf myarch.tar mydir
mydir/
mydir/reports/
mydir/reports/weather
mydir/reports/monday
mydir/reports/friday
mydir/mymeeting
mydir/party
```

The user can later extract the directories from the tape using the **x** option. The **xf** option extracts files from an archive file or device. The tar extraction operation generates all subdirectories. In the next example, the **xf** option directs **tar** to extract all the files and subdirectories from the tar file **myarch.tar**:

```
$ tar xvf myarch.tar
mydir/
mydir/reports/
mydir/reports/weather
mydir/reports/monday
mydir/reports/friday
mydir/mymeeting
mydir/party
```

You use the **r** option to add files to an already created archive. The **r** option appends the files to the archive. In the next example, the user appends the files in the **letters** directory to the **myarch.tar** archive. Here, the directory **mydocs** and its files are added to the **myarch.tar** archive:

```
$ tar rvf myarch.tar mydocs
mydocs/
mydocs/doc1
```

If you change any of the files in your directories you previously archived, you can use the **u** option to instruct tar to update the archive with any modified files. The **tar** command compares the time of the last update for each archived file with those in the user's directory and copies into the archive any files that have been changed since they were last archived. Any newly created files in these directories are also added to the archive. In the next example, the user updates the **myarch.tar** file with any recently modified or newly created files in the **mydir** directory. In this case, the **gifts** file was added to the **mydir** directory.

```
tar uvf myarch.tar mydir
mydir/
mydir/gifts
```

If you need to see what files are stored in an archive, you can use the **tar** command with the **t** option. The next example lists all the files stored in the **myarch.tar** archive:

```
tar tvf myarch.tar
drwxr-xr-x root/root 0 2000-10-24 21:38:18 mydir/
drwxr-xr-x root/root 0 2000-10-24 21:38:51 mydir/reports/
-rw-r--r-- root/root 22 2000-10-24 21:38:40 mydir/reports/weather
```

```
-rw-r--r-- root/root 22 2000-10-24 21:38:45 mydir/reports/monday
-rw-r--r-- root/root 22 2000-10-24 21:38:51 mydir/reports/friday
-rw-r--r-- root/root 22 2000-10-24 21:38:18 mydir/mymeeting
-rw-r--r-- root/root 22 2000-10-24 21:36:42 mydir/party
drwxr-xr-x root/root 0 2000-10-24 21:48:45 mydocs/
-rw-r--r-- root/root 22 2000-10-24 21:48:45 mydocs/doc1
drwxr-xr-x root/root 0 2000-10-24 21:54:03 mydir/
-rw-r--r-- root/root 22 2000-10-24 21:54:03 mydir/gifts
```

To back up the files to a specific device, specify the device as the archive. For a floppy disk, you can specify the floppy drive. Be sure to use a blank floppy disk. Any data previously placed on it will be erased by this operation. In the next example, the user creates an archive on the floppy disk in the **/dev/fd0** device and copies into it all the files in the **mydir** directory:

```
$ tar cf /dev/fd0 mydir
```

To extract the backed-up files on the disk in the device, use the **xf** option:

```
$ tar xf /dev/fd0
```

If the files you are archiving take up more space than would be available on a device such as a floppy disk, you can create a tar archive that uses multiple labels. The **M** option instructs tar to prompt you for a new storage component when the current one is filled. When archiving to a floppy drive with the **M** option, tar prompts you to put in a new floppy disk when one becomes full. You can then save your tar archive on several floppy disks.

```
$ tar cMf /dev/fd0 mydir
```

To unpack the multiple-disk archive, place the first one in the floppy drive and then issue the following **tar** command using both the **x** and **M** options. You are then prompted to put in the other floppy disks as they are needed.

```
$ tar xMf /dev/fd0
```

The **tar** operation does not perform compression on archived files. If you want to compress the archived files, you can instruct tar to invoke the gzip utility to compress them. With the lowercase **z** option, tar first uses gzip to compress files before archiving them. The same **z** option invokes **gzip** to decompress them when extracting files.

```
$ tar czf myarch.tar.gz mydir
```

Remember, a difference exists between compressing individual files in an archive and compressing the entire archive as a whole. Often, an archive is created for transferring several files at once as one tar file. To shorten transmission time, the archive should be as small as possible. You can use the compression utility gzip on the archive tar file to compress it, reducing its size, and then send the compressed version. The person receiving it can decompress it, restoring the tar file. Using gzip on a tar file often results in a file with the extension **.tar.gz**. The extension **.gz** is added to a compressed gzip file. The next example creates a compressed version of **myarch.tar** using the same name with the extension **.gz**:

```
$ gzip myarch.tar
$ ls
$ myarch.tar.gz
```

If you have a default device specified, such as a tape, and you want to create an archive on it, you can simply use **tar** without the **f** option and a device or filename. This can be helpful for making backups of your files. The name of the default device is held in a file called **/etc/default/tar**. The syntax for the **tar** command using the default tape device is shown in the following example. If a directory name is specified, all its subdirectories are included in the archive.

```
$ tar option directory-and-file-names
```

In the next example, the directory **mydir** and all its subdirectories are saved on a tape in the default tape device:

```
$ tar c mydir
```

In this example, the **mydir** directory and all its files and subdirectories are extracted from the default tape device and placed in the user's working directory:

```
$ tar x mydir
```

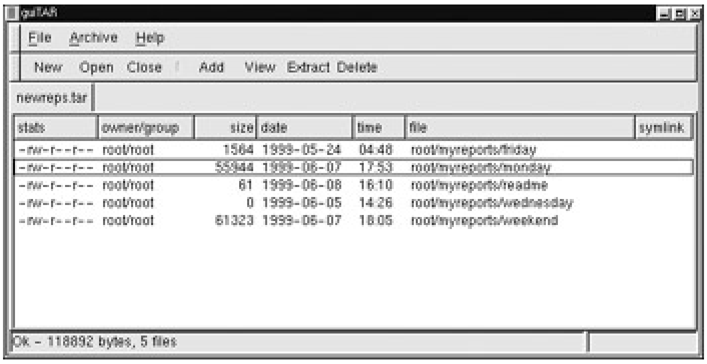
Note There are other archive programs you can use such as cpio and shar. However, tar is the one most commonly used for archiving application software.

Midnight Commander (Gnome) and konqueror (KDE)

Both file managers in Gnome and the K Desktop have the capability to display the contents of a tar archive file automatically. The contents are displayed as though they were files in a directory. You can list the files as icons or with details, sorting them by name, type, or other fields. You can even display the contents of files. Clicking a text file opens it with a text editor, and an image is displayed with an image viewer. If the file manager cannot determine what program to use to display the file, it prompts you to select an application. Both file managers can perform the same kind operation on archives residing on remote file systems, such as tar archives on FTP sites. You can obtain a listing of their contents and even read their **readme** files. The Midnight Commander file manager (Gnome) can also extract an archive. Right-click the Archive icon and select Extract.

Desktop Archivers: guiTAR, gnochive, LnxZip, Ark, KArchive, and Xtar

Several desktop applications provide a GUI interface for creating and extracting archives. These archivers provide simple methods for managing archives, enabling you to select files and set options easily. The guiTAR archiver is Gnome-based. When creating an archive, you can choose from several compression methods, including tar, rar, and zip. You can open an archive with a drag-and-drop operation, dragging an archive file from the file manager window to the guiTAR window (see Figure 32-4). Files listed in an archive can be sorted by different fields by clicking the buttons across the top of the list. gnochive is a Gnome front end for Linux archive tools. LnxZip is another Gnome front end for Linux archive tools like zip, gzip, tar, arj, and bzip2. LnxZip has the added capability of generating RPM packages.



stats	owner/group	size	date	time	file	symlink
-rw-r--r--	root/root	1564	1999-05-24	04:48	root/myreports/friday	
-rw-r--r--	root/root	55944	1999-06-07	17:53	root/myreports/monday	
-rw-r--r--	root/root	61	1999-06-08	16:10	root/myreports/readme	
-rw-r--r--	root/root	0	1999-06-05	14:26	root/myreports/wednesday	
-rw-r--r--	root/root	61323	1999-06-07	18:05	root/myreports/weekend	

Ok - 118892 bytes, 5 files

Figure 32-4: The guiTAR Gnome archiver

Ark and KArchive are K Desktop archivers. With Ark, to open a new archive, you enter a name with the **.tar.gz** extension. Once an archive is open, you can add to it by dragging files from a file manager window to the Ark window. To extract an archive, first open it and then select Extract. KArchiver lets you create archives by simply dragging files to the KArchive icon. With KArchiver, you can also search for archives and convert between gzip and bzip2 versions. You can specify a root directory or create a tape profile listing specific files and directories. The Xtar archiver is an X Window System application that can run on any file manager and provides much the same functionality as the other archivers. Once you select the tar archive to open, all the files making up the tar archive are then listed in the main window. With Xtar, you have the option of either unpacking the entire tar archive or only a few files within it. Options also has a View item for displaying short text files, such as a **readme** file.

File Compression: gzip, bzip2, and zip

Several reasons exist for reducing the size of a file. The two most common are to save space or, if you are transferring the file across a network, to save transmission time. You can effectively reduce a file size by creating a compressed copy of it. Anytime you need the file again, you decompress it. Compression is used in combination with archiving to enable you to compress whole directories and their files at once. Decompression generates a copy of the archive file, which can then be extracted, generating a copy of those files and directories.

Several compression utilities are available for use on Linux and Unix systems. Most software for Linux systems use the GNU gzip and gunzip utilities. The gzip utility compresses files and gunzip decompresses them. To compress a file, enter the command **gzip** and the filename. This replaces the file with a compressed version of it, with the extension **.gz**.

```
$ gzip mydata
$ ls
mydata.gz
```

To decompress a gzip file, use either **gzip** with the **-d** option or the command **gunzip**. These commands decompress a compressed file with the **.gz** extension and replace it with a decompressed version with the same root name, but without the **.gz** extension. When you use gunzip, you needn't even type in the **.gz** extension. **gunzip** and **gzip -d** assume it. Table 32-21 lists the different gzip options.

```
$ gunzip mydata.gz
$ ls
mydata
```

Suppose you want to display or print the contents of a compressed file without first having to decompress it. The command **zcat** generates a decompressed version of a file and sends it to the standard output. You can then redirect this output to a printer or display a utility such as more. The original file remains in its compressed state.

```
$ zcat mydata.gz | more
```

You can also compress archived tar files. This results in files with the extensions **.tar.gz**. Compressed archived files are often used for transmitting extremely large files across networks.

```
$ gzip myarch.tar
$ ls
myarch.tar.gz
```

You can compress tar file members individually using the **tar z** option that invokes **gzip**. With the **z** option, tar invokes gzip to compress a file before placing it in an archive. Archives with members compressed with the **z** option, however, can neither be updated nor added to. All members must be compressed and all must be added at the same time.

Table 32-21: The gzip Options

Option	**Execution**
-c	Sends compressed version of file to standard output; each file listed is separately compressed: **gzip** -c mydata preface myfiles.gz
-d	Decompresses a compressed file; or, you can use **gunzip**: **gzip** -d myfiles.gz **gunzip** myfiles.gz
-h	Displays help listing.
-l *file-list*	Displays compressed and uncompressed size of each file listed: **gzip** -l myfiles.gz.
-r *directory-name*	Recursively searches for specified directories and compresses all the files in them; the search begins from the current working directory; when used with **gunzip**, compressed files of a specified directory are uncompressed.
-v *file-list*	For each compressed or decompressed file, displays its name and the percentage of its reduction in size.
-*num*	Determines the speed and size of the compression; the range is from -1 to -9. A lower number gives greater speed but less compression, resulting in a larger file that compresses and decompresses quickly; -1 gives the quickest compression, but with the largest size; -9 results in a very small file that takes longer to compress and decompress. The default is –6.

You can also use the **compress** and **uncompress** commands to create compressed files. They generate a file that has a **.Z** extension and use a different compression format than gzip. The **compress** and **uncompress** commands are not that widely used, but you may run across **.Z** files occasionally. You can use the **uncompress** command to decompress a **.Z** file. The gzip utility is the standard GNU compression utility and should be used instead of **compress**.

Another popular compression utility is bzip2. It compresses files using the Burrows-Wheeler block-sorting text compression algorithm and Huffman coding. The command line options are similar to gzip by design, but they are not exactly the same. See the bzip2 Man page for a complete listing. You compress files using the **bzip2** command and decompress with **bunzip2**. The **bzip2** command creates files with the extension **.bz2**. You can use **bzcat** to output compressed data to the standard output. The **bzip2** command compresses files in block and enables you to specify their size (larger blocks give you greater compression). As with gzip, you can use bzip2 to compress tar archive files. The following example compresses the **mydata** file into a bzip compressed file with the extension **.bz2**:

```
$ bzip2 mydata
$ ls
mydata.bz2
```

To decompress, use the **bunzip2** command on a bzip file.

```
$ bunzip2 mydata.bz2
```

Zip is a compression and archive utility modeled on PKZIP**,** which was used originally on DOS systems. Zip is a cross-platform utility used on Windows, Mac, MSDOS, OS/2, Unix, and Linux systems. Zip commands can work with archives created by PKZIP and PKZIP programs and can use Zip archives. You compress a file using the **zip** command. This creates a Zip file with the **.zip** extension. If no files are listed, **zip** outputs the compressed data to the standard output. You can also use the **-** argument to have **zip** read from the standard input. To compress a directory, you include the **-r** option. The first example archives and compresses a file:

```
$ zip mydata
$ ls
mydata.zip
```

The next example archives and compresses the **reports** directory:

```
$ zip -r reports
```

A full set of archive operations is supported. With the **-f** option, you can update a particular file in the zip archive with a newer version. The **-u** option replaces or adds files, and the **-d** option deletes files from the zip archive. Options also exist for encrypting files and DOS-to-Unix end-of-line translations, and including hidden files.

To decompress and extract the Zip file, you use the **unzip** command.

```
$ unzip mydata.zip
```

Journaling

With release 7.2, Red Hat introduced journaling capabilities with the ext3 file system. Journaling provides for fast and effective recovery in case of disk crashes, instead of using fsck or e2fsck. With journaling a log is kept of all file system actions. These are placed in a journal file. In the event of a crash, Linux only needs to read the journal file to restore the system to its previous state. Files that were in the process of writing to the disk, can be restored to their original state. Journaling also avoids lengthy fsck checks on reboot that occur when your system suddenly looses power or if it freezes and has to be restarted physically. Your system just reads its journal files to restore the file system, instead of manually checking each file and directory with fsck.

Journaling is implemented automatically with ext3. The ext3 file system is also fully compatible with the earlier ext2 version it replaces. To create an ext3 file system you use the **mk2fs** command with the **–j** option. You can even upgrade ext2 file systems to ext3 versions automatically, with no loss of data or change in partitions. This upgrade just adds a journal file to an ext2 file system and enables journaling on it, using the **tune2fs** command. Be sure to change the ext2 file type to ext3 in any corresponding /etc/fstab entries. The following example converts the ext2 file system on /dev/hda3 to an ext3 file system by adding a journal file (**-j**).

```
tune2fs –j /dev/hda3
```

The ext3 filesystem has three journalling modes of operation: ordered, journal, and writeback. The ordered mode is the default, guaranteeing the integrity of files written recently. The writeback mode works faster, but performs less logging and may result in corrupt data in the case of crash. The journal mode is the slowest, but copies all data to the journal, allowing for complete recovery. You set the mode with the data option in either the **mount** command or **fstab** entry, as in **data=journal**.

There are other kind of journaling file systems you can use on Linux. These include ReiserFS, JFS, and XFS. ReiserFS is named after Hans Reiser and provides a completely reworked file system structure based on journaling (www.reiserfs.org). JFS is the IBM version of a journaling file system, designed for use on servers providing high throughput such as ebusiness enterprise servers (oss.software.ibm.com/ developerworks/opensource/jfs/). It is free distributed under the GNU public license. XFS is another high performance journaling system developed by Silicon Graphics (oss.sgi.com/projects/xfs/). XFS is compatible with RAID and NFS file systems.

Though journaling is often used to recover from disk crashes, a journal-based file system can do much more. The ext3, JFS, and XFS file systems only provide the logging operations used in recovery, whereas ReiserFS uses journaling techniques to completely rework file system operations. In ReiserFS journaling is used to read and write data, abandoning the block structure used in traditional Unix and Linux systems. This gives it the capability to access a large number of small files very quickly, as well as use only the amount of disk space they would need. However, efficiency is not that much better with larger files.

Chapter 33: Devices and Printers

Overview

All devices, such as printers, terminals, and CD-ROMs, are connected to your Linux operating system through special files called *device files.* Such a file contains all the information your operating system needs to control the specified device. This design introduces great flexibility. The operating system is independent of the specific details for managing a particular device; the specifics are all handled by the device file. The operating system simply informs the device what task it is to perform, and the device file tells it how. If you change devices, you only have to change the device file, not the whole system.

To install a device on your Linux system, you need a device file for it, software configuration such as provided by a configuration tool, and kernel support—usually supplied by a module or already built into the kernel. An extensive number of device files are already set up for different kinds of devices. You usually only need to choose one of these. For kernel support, you may have to load a kernel module or recompile the kernel, both simple procedures. In most cases, support is already built into the kernel. Configuration of your device may be provided by desktop configuration tools such as the Gnome Control Center, system configuration tools such as Linuxconf, or a module configuration interface such as that provided for sound modules by sndconfig.

Device Files

The name of a device file is designed to reflect the task of the device. Printer device files begin with **lp** for "line print." Because you could have more than one printer connected to your system, the particular printer device files are distinguished by two or more numbers or letters following the prefix **lp**, such as **lp0**, **lp1**, **lp2**. The same is true for terminal device files. They begin with the prefix **tty**, for "teletype," and are further distinguished by numbers or letters such as **tty0**, **tty1**, **ttyS0**, and so on. You can obtain a complete listing of the current device filenames and the devices for which they are used from the **kernel.org** Web site at **http://www.kernel.org/pub/linux/docs/device-list/devices.txt**.

All of these filenames will be implemented as device files in your **/dev** directory. Here you can find printer, CD-ROM, hard drive, SCSI, and sound device files, along with many others. Certain link files bear common device names that are often linked to the actual device file used. For example, a **/dev/cdrom** symbolic link links to the actual device used for your CD-ROM. If your CD-ROM is an IDE device, it may use the device file **hdc**. In this case, **/dev/cdrom** would be a link to **/dev/hdc**. In effect, **/dev/cdrom** is another name for **/dev/hdc**. You can use **/dev/cdrom** to reference your CD-ROM's device file, instead of **/dev/hdc**. A **/dev/modem** link file also exists for your modem. If your modem is connected to the second serial port, its device file would be **/dev/ttyS1**. In this case, **/dev/modem** would be a link to that device file. Applications can then use **/dev/modem** to access your modem, instead of having to know the actual device file used. A listing of commonly used device links is shown in Table 33-1.

Table 33-1: Device Links

Link	Description
/dev/mouse	Current mouse device
/dev/tape	Current tape device
/dev/cdrom	Current CD-ROM device
/dev/cdwriter	Current CD-writer device
/dev/scanner	Current scanner device
/dev/modem	Current dial-out device, modem port
/dev/root	Current root file system
/dev/swap	Current swap device

Note You will notice that there are no entries for the Ethernet devices in the **/dev** file, such as **eth0** or **eth1**. That is because these are really aliases for kernel modules defined in the **/etc/modules.conf** file. They are not device files.

Two types of devices are implemented in Linux: block and character. A *block device,* such as a hard disk, transmits data a block at a time. A *character device,* such as a printer or modem, transmits data one character at a time, or rather as a continuous stream of data, not as separate blocks. Device driver files for character devices have a *c* as the first character in the permissions segment displayed by the **ls** command. Device driver files for block devices have a *b.* In the next example, **lp0** (the printer) is a character device and **hda1** (the hard disk) is a block device:

```
# ls -l hda1 lp0
brw-rw---- 1 root disk 3, 1 Sep 7 1994 hda1
crw-r----- 1 root daemon 6, 0 Dec 31 1979 lp0
```

Although most distributions include an extensive set of device files already set up for you, you can create your own. You use the **mknod** command to create a device file, either a character or block type. The **mknod** command has the following syntax:

```
mknod options device device-type major-num minor-num
```

The device type can be either *b, c, p,* or *u.* As already mentioned, the *b* indicates a block device, and *c* is for a character device. The *u* is for an unbuffered character device, and the *p* is for a FIFO device. Devices of the same type often have the same name; for example, serial interfaces all have the name: **ttyS**. Devices of the same type are then uniquely identified by a number attached to the name. This number has two components: the major number and the minor number. Devices may further have the same major number, but if so, the minor number is always different. This major and minor structure is designed to deal with situations in which several devices may be dependent on one larger device, such as several modems connected to the same I/O card. All would have the same major number that would reference the card, but each modem would have a unique minor number. Both the minor and major numbers are required for block and character devices (*b, c,* and *u*). They are not used for FIFO devices, however.

For example, Linux systems usually provide device files for three parallel ports (**lp0–2**). If you need more, you can use the **mknod** command to create a new one. Printer devices are character devices and must be owned by the root and daemon. The permissions for printer devices are read and write for the owner and the group, 660 (see Chapter 12 for a discussion of file permissions). The major device number is set to 6, while the minor device number is set to the port number of the printer, such as 0 for LPT1 and 1 for LPT2. Once the device is created, you use **chown** to change its ownership to **root.daemon**. In the next example, a parallel printer device is made on a fourth parallel port, **/dev/lp3**. The **-m** option specifies the permissions—in this case, 660. The device is a character device, as indicated by the *c* argument following the device name. The major number is 6, and the minor number is 3. If you were making a device at **/dev/lp4**, the major number would still be 6, but the minor number would be 4. Once the device is made, the **chown** command then changes the ownership of the parallel printer device to **root.daemon**. Be sure to check if a spool directory has been created for your device. If not, you need to make one.

```
# mknod -m 660 /dev/lp3 c 6 3
# chown root.daemon /dev/lp3
```

Valid device names along with their major and minor numbers are listed in the **devices.txt** file located in the documentation directory for the kernel source code, **/usr/src/linux**. When creating a device, you use the major and minor numbers as well as the device name prefix for the particular kind of device you are creating. Most of these devices are already created for you and are listed in the **/etc/dev** directory.

Device Information: /proc

The **/proc** file system (see Chapter 32) maintains special information files for your devices. The **/proc/devices** file lists all your installed character and block devices along with their major numbers. IRQs, DMAs, and I/O ports currently used for devices are listed in the **interrupts**, **dma**, and **ioports** files, respectively. Certain files list information covering several devices, such as **pci**, which lists all your PCI devices, and **sound**, which lists all your sound devices. The **sound** file will list detailed information about your sound card. Several subdirectories, such as **net**, **ide**, and **scsi**, contain information files for different devices. Table 33-2 lists several device-related **/proc** files (see Chapter 32 for other entries).

Table 33-2: /Proc Device Information Files

File	Description
/proc/devices	List of the device drivers configured for the currently running kernel
/proc/dma	Displays the DMA channels currently used
/proc/interrupts	Displays the IRQs (interrupts) in use
/proc/ioports	Shows the I/O ports in use
/proc/pci	Lists PCI devices
/proc/sound	Lists sound devices
/proc/scsi	Directory for SCSI devices
/proc/ide	Directory for IDE devices
/proc/net	Directory for network devices

Installing and Managing Printers: LPRng

Printers are installed and managed on Red Hat by the Linux print server, next generation (LPRng). LPRng is an enhanced version of the Berkeley Line Printer Daemon (LPD) **lpd** and associated **lpr** applications. It features a wide range of capabilities that include security measures and access to remote printers. Many of the commands are the same as those used by LPD on a standard Unix system. The Linux printer server program is called **lpd**, the line printer daemon. Printers are installed to run under **lpd**, which then handles print jobs for them both locally and from remote sources. Though **lpd** is called the line printer daemon, it is designed to manage any kind of printer, not just line printers. You should think of it as a general-purpose print server capable of handling laser, inkjet, postscript, and dot-matrix printers. You can find out more information about printing in Linux at **www.linuxprinting.org**. LPRng also features a companion IFHP filter package which provides hardware-level support for postscript, PCL, text printers, among others (**www.astart.com/lprng**).

Printers can be attached directly to your system or attached to other systems on your network. A printer attached directly to your system is referred to as a local printer. For example, a local printer would be a printer connected to your parallel port on your PC. A remote printer would be one connected directly to another system on your network. A remote printer could also be one that is designed to operate as its own host on a network, accessible directly by other systems on the network. You could even have printers connected to a single system operating just as a print server that then manages access to them. Access to both local and remote printers is managed by the **lpd** daemon. Request are submitted to the **lpd** daemon along with the print job. The **lpd** daemon then spools the print job, making its own copy, and then sends that job to the specified printer.

The **lpd** daemon was installed and configured on your Linux system when you installed Red Hat. **lpd** is run as a stand-alone process by the lpd startup script in the **/etc/rc.d/init.d** directory. You can use the **service** command on this script to start, stop, and restart the daemon.

```
service lpd restart
```

lpd makes use of two configuration files: **lpd.conf** and **lpd.perms**. **lpd.conf** contains general **lpd** configuration commands. You use **lpd.perms** to set up rules by which you can restrict access to the **lpd** server. Here you can deny access by certain hosts, users, or even networks.

Requests to print documents are performed by print clients like **lpr**. When a document is submitted for printing, it becomes a print job that is placed on a queue for the printer it was sent to. While the job is on the queue waiting to print, you can check its status and even remove it from the queue, canceling the job. The **lpq** client lets you check a print queue, **lpc** allows you to make changes to it, and **lprm** can be used to remove a print job from a queue.

Installing Printers

To use any printer, it first has to be installed on a Linux system that is on your network. A local printer is installed directly on your own system. This involves creating an entry for the printer in the **/etc/printcap** file that defines the kind of printer it is along with other features like the device file and spool directory it uses. Installing a printer is fairly simple: determine which device file to use for the printer, and place printer configuration entries for it in your **/etc/printcap** file. You can use several configuration tools to enable you to set up and configure your printer easily. Red Hat systems provide the printconf utility discussed here. KDE also provides the K Printer System utility.

Red Hat Linux creates three device names for parallel printers automatically during installation: **lp0**, **lp1**, and **lp2**. (Most systems currently use **lp1**.) The number used in these names corresponds to a parallel port on your PC. **lp0** references the LPT1 parallel port usually located at address 0x03bc, **lp1** references the LPT2 parallel port located at 0x0378, and **lp2** references LPT3 at address 0x0278. If you are unsure at what address your parallel port is located, you can use the **msd.exe** command on your DOS system to find it. Serial printers use the serial ports, referenced by the device files **ttyS0**, **ttyS1**, **ttyS2**, and so on**.** For a detailed explanation of printer installation, see the **Printing-HOWTO** file in **/usr/share/doc/HOWTO**.

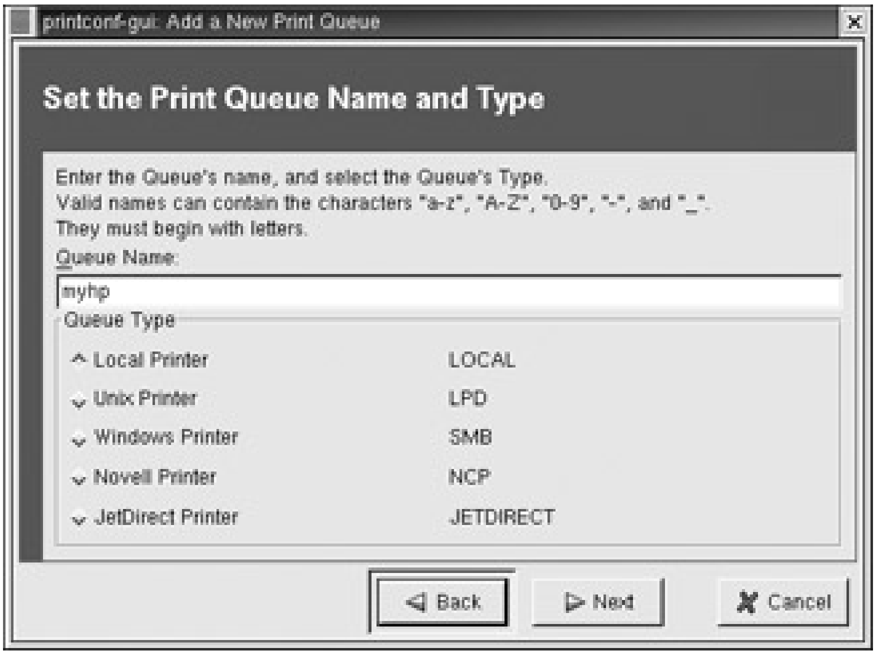
printconf

The printconf utility provided on Red Hat distributions is an easy interface for setting up and managing your printers. Using only printconf, you can easily install a printer on your Linux system. You can start printconf by selecting the Printer Configuration entry in the Gnome System menu. The printconf utility enables you to select the appropriate driver for your printer, as well as to set print options such as paper size and print resolutions. Once you have configured your printer with printconf, printconf will generate an entry for it in the **/etc/printcap** file. See the printconf section in the Red Hat Customization Guide for more details.

When you start up printconf, you are presented with a window that lists your installed printers. To add a new printer, click the New button. To edit an installed printer, double-click its entry or select it and click the Edit button. Once you have made your changes, you can click on the Apply button to save your changes and restart the printer daemon. If you have more than one printer on your system, you can make one the default by selecting it and then clicking on the Default button. The Delete button will remove a printer configuration. You can test your printer with a PostScript, A4, or ASCII test sheet selected from the Test menu.

When you select New a series of dialogs will be displayed where you can enter the printer name, its type, and its driver. You can also edit a printer to change any settings. When editing, a set of four tabbed panes are displayed for the printer name, queue type, driver, and options.

For the queue selection, you can specify entries for the printer device and spool directory. The device is the port to which the printer is connected. For the first three parallel ports, these are **lp0**, **lp1**, and **lp2;** for serial ports, these are **ttyS0**, **ttyS1**, and **ttyS2;** and so on (see Figure 33-1). From a drop-down menu, you can also specify whether the printer is local or remotely connected through a Linux/Unix, Windows (SMB), or NetWare network.



Set the Print Queue Name and Type

Enter the Queue's name, and select the Queue's Type.
Valid names can contain the characters "a-z", "A-Z", "0-9", "-", and "_".
They must begin with letters.

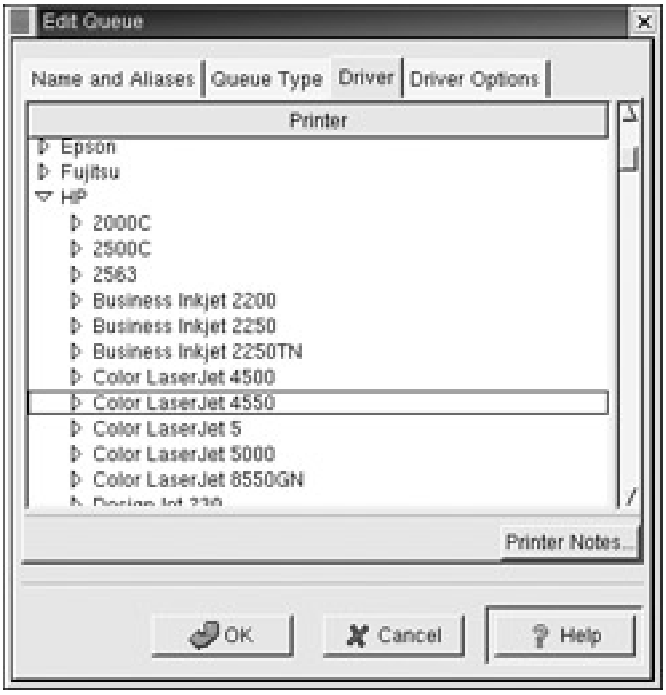
Queue Name: myhp

Queue Type

● Local Printer	LOCAL
○ Unix Printer	LPD
○ Windows Printer	SMB
○ Novell Printer	NCP
○ JetDirect Printer	JETDIRECT

Figure 33-1: printconf printer queue types

For the driver selection, you are presented with an expandable tree of printer types. You first select the manufacturer, such as Canon or Apple, which then expands to a list of particular printer models (see Figure 33-2). Click on yours.



Printer
- Epson
- Fujitsu
- HP
  - 2000C
  - 2500C
  - 2563
  - Business Inkjet 2200
  - Business Inkjet 2250
  - Business Inkjet 2250TN
  - Color LaserJet 4500
  - Color LaserJet 4550
  - Color LaserJet 5
  - Color LaserJet 5000
  - Color LaserJet 8550GN
  - DesignJet 230

Figure 33-2: printconf printer drivers

For the options selection, you can specify printer features such as paper size and resolution. When you finish, click OK to close the window. You then see your printer listed in the printconf window, as shown in Figure 33-1. Choose the Quit item from the File menu to quit printconf. You are now ready to print. For a detailed explanation of printer installation, see the **Printing-HOWTO** file in **/usr/share/doc/HOWTO**.

Note If you cannot find the drivers for your printer, you can may be able to download them from **www.linuxprinting.org**. The site maintains an extensive listing of drivers.

Setting Up Printers with Linuxconf and Webmin

To set up a printer using Linuxconf, select Add | Edit printers under the Miscellaneous Services | Printer menu in the Config panel (be sure you are using the full version of Linuxconf). This opens a Printer Setup panel listing all the printers installed on your system. Click the Add button to open an Adding a Printer panel where you can enter the printer name and specify whether it is a local or remote printer. Once the printer is added, you can then double-click its entry in the Printer Setup panel to open a Printer Properties panel with panels for General, Printertype, and Filter options. In the Printertype panel, you select the device file to use for this printer, such as **/dev/lp0**. In the Filter options panel, you can choose options such as resolution, paper size, and color depth. To select your printer type, click the Filter button to open a listing of printers to choose from.

To manage printers with Webmin, you select Printer Administration from the Hardware panel. To add a printer, click the Add a New Printer link on the Printer Administration page to display the Create Printer page. Here you can enter the printer name and description as well as enable the printer and specify the port. For a local printer, you can then select the appropriate driver.

Printer Devices and /etc/printcap

When your system prints a file, it makes use of special directories called *spool directories.* A print job is a file to be printed. When you print a file to a printer, a copy of it is made and placed in a spool directory set up for that printer. The location of the spool directory is obtained from the printer's entry in the **/etc/printcap** file. On Red Hat Linux, the spool directory is located at **/var/spool/lpd** under a directory with the name of the printer. For example, the spool directory for the myepson printer would be located at **/var/spool/lpd/myepson**. The spool directory contains several files for managing print jobs. Some will bear as their extension the name of the printer. For example, the myepson printer will have the files **control.myepson** (provides printer queue control) and **active.myepson** (for the active print job), as well as **log.myepson** (the log file).

The **/etc/printcap** file holds entries for each printer connected to your system. A *printcap entry* holds information, such as the pathname for a printer's spool directory and the device name of the printer port the printer uses. The first field in a **printcap** entry is a list of possible names for the printer. These are names you can make up yourself, and you can add others if you want. Each name is separated by a |. You use these names to identify the printer when entering various printer commands or options, such as the **-P** option. These names are also used for special shell variables, such as the **PRINTER** variable, used in many initialization scripts.

Note If you manage your printers with printconf, then printconf will maintain and overwrite the **/etc/printcap** file as you change and add printers. Should you need to create a printcap entry manually, you can place it in the **/etc/printcap.local** file.

The fields following the list of names set different fields for your printer. The fields have two-letter names and are usually assigned a value using =. These assignments are separated by colons. Three of the more important fields are **lp**, **sd**, and **of**. The **lp** field is set to the device name the printer uses. The **sd** field is set to the pathname of the spool directory, and **of** is set to the particular filter used for this printer. Some have Boolean values and simply list the field name with no assignment for a true value. You can find a complete listing of the **printcap** fields in the **printcap** Man pages: **man printcap**. An example of a **printcap** entry follows.

```
myprinter|myepson:\
        :sh:\
        :ml=0:\
        :mx=0:\
        :sd=/var/spool/lpd/myprinter:\
        :lp=/dev/lp0:\
        :lpd_bounce=true:\
        :if=/usr/share/printconf/mf_wrapper:
```

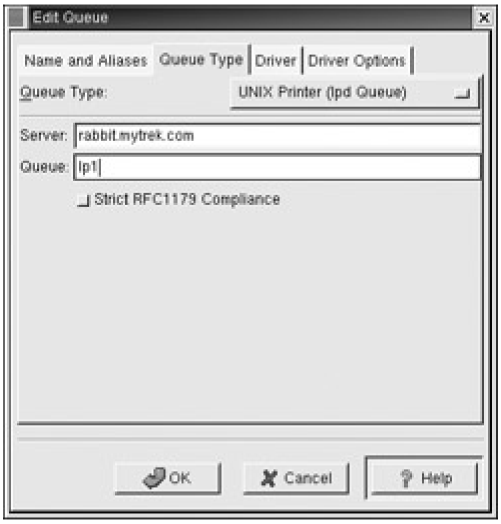
Tip Instead of making your own entries in the **/etc/printcap** file, you can use a printer configuration tool such as the printconf utility, Linuxconf, or Webmin to make them for you automatically.

Installing Remote Printers

To install a remote (network or remote host–attached) printer, you place remote entries for the printer host and device in the printer's **/etc/printcap** file entry. An **:rm** entry identifies the remote host that controls the remote printer, and an **:rp** entry specifies the device name of the remote printer. In the following example, the remote printer is located at **rabbit.mytrek.com** and is called **lp1**:

```
:rm=rabbit.mytrek.com
:rp=lp1
```

You can also use printconf to set up a remote printer on Linux, Unix, Microsoft, or Novell networks (see Table 33-3). When you add a new printer, an edit window opens with a sidebar for selecting printer configuration tasks. For the queue, a pane is displayed with a drop-down menu where you can select whether this is a local, Unix (lpd share), Windows (SMB share), Novell (NDP queue), or JetDirect printer. For a remote Linux or Unix printer, select Unix Printer (lpd share). This displays a dialog box for configuring the remote printer with entries for the server and the queue (see Figure 33-3). For the server, enter the host name for the system that controls the printer. For the queue, enter the device name on that host for the printer. A Novell (NCP queue) screen will add entries for the user and the password. A Windows (SMB share) will have entries for the share name, host IP address, workgroup, user, and password. Check the Red Hat Customization Guide for more details.



Queue Type: UNIX Printer (lpd Queue)

Server: rabbit.mytrek.com

Queue: lp1

□ Strict RFC1179 Compliance

Figure 33-3: printconf remote printer configuration

Table 33-3: printconf Supported Networks for Remote Printers

Printer	Description
Local Printer	Any printer that is directly connected to your system.
Unix Printer (lpd Spool)	A printer connected to another Linux or Unix system on your network (TCP/IP).
Windows Printer (SMB Share)	A printer connected to another system that is on an SMB network such as a Microsoft Windows system. Samba must be enabled.
Novell Printer (NCP Queue)	A printer connected to another system that is on a Novell's NetWare network.
JetDirect Printer	A printer that is connected to the network directly instead of through another system.

A Windows (SMB share) printer is one located on a Windows network. To access an SMB share remote printer, you need to install Samba and have the Server Message Block services enabled using the **smbd** daemon. Printer sharing must, in turn, be enabled on the Windows network. In the printconf SMB configuration panel, you need to enter the name of the share, its IP address, the name of the printer's workgroup, and the user name and password. The share is the hostname and printer name in the format \\hostname\printername. The hostname is the computer where the printer is located, and printer name is the name of the printer as it is known to remote hosts. The user name and password can be one for the printer resource itself, or for access by a particular user. You can then use a print client **lpr** to print a file to the Windows printer. **lpr** will invoke the Samba client **smbclient** to send the print job to the Windows printer.

Note On Linuxconf, you can also set up a remote printer. When adding a printer, you determine if it is a local, remote, SMB/Windows, or NetWare printer. When you open the Printer Properties panel for that printer, the Printertypes options panel will display entry boxes where you can enter the remote host and the remote queue for that printer.

Configuring lpd

LPRng allows you to configure your **lpd** server, providing support found in other servers such as secure access or setting global defaults. There are only two configuration files to manage, and both are heavily commented. Red Hat provides basic versions for both. An extensive set of features are available, letting you create servers with powerful and complex capabilities.

General Configuration: lpd.conf

The general configuration for the **lpd** print server is handled in the **/etc/lpd.conf** file. Here you can specify features that apply to all printers and print management. You should think of these more as default features, as any of them can be overridden in a printer's **printcap** entry.

The **lpd.conf** file installed with LPRng on Red Hat will hold an extensive list of configuration parameters. They will all be commented out, using preceding # signs. A comment describing the parameter precedes each. The entry itself lists the default value given to the parameters, preceding the entry with the term "default". The entry for the connect_timeout parameter is shown here:

```
# Purpose: connection timeout for remote printers
# default connect_timeout=10 (INTEGER)
```

To create your own timeout entry, it is best to add your own entry below, as shown here. Notice that the preceding # and the term "default" are missing from the new entry. The connection timeout for remote printers is now set at 20. This can be overridden by a printer's **printcap** file entry.

```
# Purpose: connection timeout for remote printers
# default connect_timeout=10 (INTEGER)
connect_timeout=20
```

Parameters can also be flags that you can turn on or off. An off flag is noted with an attached @ sign. In the following example, the allow_user_logging parameter is a flag that will allow users to request login information. By default it is turned off. The following example turns it on:

```
# Purpose: allow users to request logging info using lpr -mhost%port
# default allow_user_logging@ (FLAG off)
allow_user_logging
```

Printer Security: lpd.perms

LPRng provides access control rules for controlling access to your print server, and thereby the printers it controls, by remote users. These are placed in the **lpd.perms** file and can be used to refuse print services to specific hosts or users. Such rules consist of an action and a set of keys (see Table 33-4). The keys specify criteria to be met, and if met, their associated action is taken. In each rule you have one action and one or more keys. If there are several keys, all their criteria must be met for the action to take place. An example of a key would be to specify the IP address of a host. In addition, you have to specify the kind of service that is being requested, such as a printing or connection request. The action is usually either ACCEPT or REJECT. Keys operate as flags or variables. To specify a host you would assign the address to the HOST key, using an assignment operator. For a user, you would use USER, and for IP addresses, IFIP. The kind of service is specified by the SERVICE key. For a printing request, the value you assign is the **lpd** code **P**. The following example specifies a host as the key and will REJECT any request from that address:

Table 33-4: lpd.perms Keys

Key	**Description**
SERVICE	Type of service
USER	Users specified in a print job control file
HOST	Hosts specified in a print job control file
IP	IP address specified in a print job control file
IFIP	IP address of requests origin (interface IP address)
GROUP	Check to see if user is part of specified group
REMOTEHOST	Hostname or IP address of remote host specified in a printer command or derived from network information
REMOTEPORT	Port number of remote connection
REMOTEUSER	Remote user specified in a printer command or derived from network information
SAMEUSER	Checks to see if the user issuing a command for a job (REMOTEUSER) is the same as the user that created it (USER), as listed in the job's control file
SAMEHOST	Checks to see if the host issuing a command for a job (REMOTEHOST) is the same as the host that created it (HOST), as listed in the job's control file
PRINTER	Specified printer
SERVER	Request originated from print server
AUTH	Enable authentication
AUTHTYPE	Type of authentication
AUTHUSER	Authenticated userID
AUTHFROM	Authenticated ID of requestor

```
REJECT SERVICE=P HOST=192.168.0.57
```

A special DEFAULT rule is designed to match any request. You can then specify the default action to take. The following entry accepts any request. A REJECT action would reject any request.

```
DEFAULT ACCEPT
```

The **lpd.perms** file consists of a set of rules that are sequentially evaluated until a match is found. The DEFAULT action rule should be the last rule and is normally one to accept any requests. In other words, any request that is not matched by the previous rules is accepted. Normally you would set up rules to reject certain requests, like requests for specific hosts. Most requests would not match these rules and then should fall through to the DEFAULT action, which would ACCEPT them.

SERVICE key values differ depending on type of request submitted. These can range from a simple connection request by a remote server to a removal of a print job. The print clients like **lpr**, **lpq**, **lprm**, and **lpc** will make different kinds of service requests. **lpq** will make a request for queue information that has the key value Q for SERVICE. **lprm** will issue a removal request indicated by a key value M. The different service values for the SERVICE key are shown here:

Key Value	**Service Requests**
X	Connection request
R	Job spool
P	Printing
Q	Queue status information (lpq)
M	Removal request (lprm)
C	Printer control (lpc)
S	Printer status (lpc)

A further distinction is made between keys that reference information in a print job's control file and that provided by a print command like **lpc** or **lpq**. For example, the HOST key references hostnames in a print job's control file, allowing you to reject or accept print job requests. The REMOTEHOST key will reference hostnames derived from the remote connection directly for all types of requests except for an **lpr** request. Check the key table in the **lpd.perm** or **lpd** Man page for a detailed breakdown of requests and the type of information they reference. As another example, SAMEUSER will match the user indicated in a print request for a print job, such as status or removal, against the user in the control file for that print job, to make sure they are the same user.

The settings for the **/etc/lpd.perms** file included with Red Hat are shown here. The file itself includes details documentation as well as several commented recommended entries.

```
# allow root on server to control jobs
ACCEPT SERVICE=C SERVER REMOTEUSER=root
# allow anybody to get server, status, and printcap
ACCEPT SERVICE=C LPC=lpd,status,printcap
# reject all others
REJECT SERVICE=C
#
# allow same user on originating host to remove a job
ACCEPT SERVICE=M SAMEHOST SAMEUSER
# allow root on server to remove a job
ACCEPT SERVICE=M SERVER REMOTEUSER=root
REJECT SERVICE=M
# all other operations allowed
DEFAULT ACCEPT
```

The Print Queue Clients

As noted previously, printing on your system is handled by a print daemon called **lpd**, which is constantly running, waiting for print jobs and then managing their printing procedures. The **lpd** daemon places its print jobs on print queues. Once on the queue, there are several clients you can use to manage print jobs such as **lpq**, which lists a print queue, **lpc**, which can reorder it, and **lprm**, which can remove a print job, effectively canceling it. Documents are initially submitted to **lpd** with the **lpr** print command.

The **lpr** client submits a job, and **lpd** then takes it in turn and places it on the appropriate print queue. **lpr** takes as its argument the name of a file. You can also feed data to **lpr** through the standard input, piping in the data to be printed from another operation. The **-P** option enables you to specify a particular printer. In the next example, the user first prints the file **preface**. Then he or she uses the **cat** command to generate combined output of the files **intro** and **digest**. This is piped to the **lpr** command, which then prints it. Finally, the user prints the file **report** to the printer with the name **myepson**.

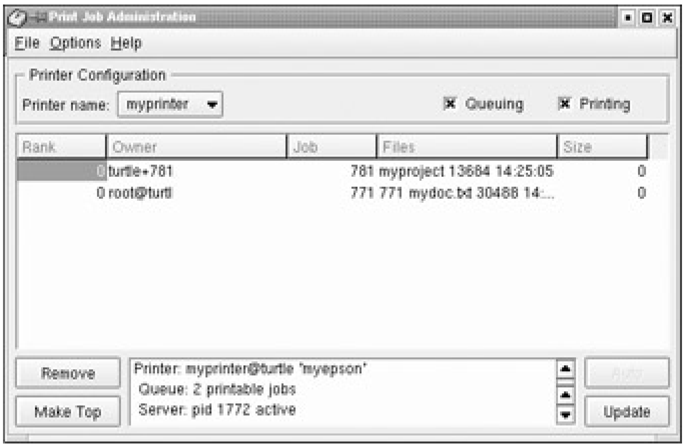
```
$ lpr preface
$ cat intro digest | lpr
$ lpr -P myepson report
```

Tip If no printer is specified for the **lpr** command, the default printer is used. This is determined by either the default_printer entry in **lpd.conf**, the PRINTER environment variable, or the first entry in the **/etc/printcap** file.

You can also print directly to the printer by simply redirecting output to the printer's device file. This does not place anything on the print queue. The print operation becomes a command to be immediately executed. However, your system is occupied until the file completes printing. The following example uses this technique to print the **report** file to a printer connected to device **lp1**:

```
$ cat report > /dev/lp1
```

To manage the printing jobs on your printer or printers, you can use either Klpq or the LPRng clients **lpc**, **lpq**, and **lprm**. Klpq is a KDE desktop utility installed with Red Hat and labeled the Print Job Administration tool (see Figure 33-4). With Klpq, you can list the print jobs for a printer, remove a print job, and move a print job to the top of the queue. You can also disable printing for a printer. To have the print queue listing automatically updated, you can set an update frequency in the Options menu.



Printer name: myprinter ☒ Queuing ☒ Printing

Rank	Owner	Job	Files	Size
0	turtle+781	781	myproject 13684 14:25:05	0
0	root@turtl	771	771 mydoc.txt 30488 14:...	0

Printer: myprinter@turtle 'myepson'
Queue: 2 printable jobs
Server: pid 1772 active

Figure 33-4: Managing a printer queue with Klpq

You can use **lpc** to enable or disable printers, reorder their print queues, and reexecute configuration files. To use **lpc**, enter the command **lpc** at the shell prompt. You are then given an **LPC>** prompt at which you can enter **lpc** commands to manage your printers and reorder their jobs. The **status** command with the name of the printer displays whether the printer is ready, how many print jobs it has, and so on. The **stop** and **start** commands can stop a printer and start it back up.

```
# lpc
lpc> status hp1
hp1|lp1:
 queuing is enabled
 printing is enabled
 1 entry in spool area
```

You can manage the print queue using the **lpq** and **lprm** commands. The **lpq** command lists the printing jobs currently on the print queue. With the **-P** option and the printer name, you can list the jobs for a particular printer. If you specify a user name, you can list the print jobs for that user. With the **-l** option, **lpq** displays detailed information about each job. If you want information on a specific job, simply use that job's ID number with **lpq**. To check the status of a printer you use **lpstat**.

With the **lprm** command, you can remove a printing job from the queue, erasing the job before it can be printed. The **lprm** command takes many of the same options as **lpq**. To remove a specific job, use **lprm** with the job number. To remove all printing jobs for a particular user, enter **lprm** with the user name. To remove all printing jobs for a particular printer, use the **-P** option with the printer name.

The **lprm** command has a special argument indicated by a dash, **-**, that references all print jobs for the user who issues the command. For example, to remove all your own print jobs, enter **lprm -**. If you logged in as the root user, **lprm -** removes all print jobs for all printers and users from the print queue, emptying it completely.

You should not use **lprm** to kill a printing job that has already started printing. Instead, you may have to use the **kill** command on the print job process. You can display processes using the **ps -ax** command, and then use **kill** and the number of the process to end it. For a job that is already printing, you see a process for its filter. This is the process to kill. If the process does not end, you can force its termination by using the –**s** option and the kill signal number, 9.

Table 33-5 shows various printer commands; Table 33-6 includes **lpc** commands.

Table 33-5: Printer Commands

Printer Management	**Description**
Klpq	KDE print queue management tool.
lpr *options file-list*	Prints a file; copies the file to the printer's spool directory, and places it on the print queue to be printed in turn.**-P***printer* prints the file on the specified printer.
lpq *options*	Displays the print jobs in the print queue.**-P***printer* prints queue for the specified printer.**-l** prints a detailed listing.
lpstat *options*	Displays printer status.
lprm *options Printjob-id* or *User-id*	Removes a print job from the print queue; you identify a particular print job by its number as listed by **lpq;** if you use *User-id*, it removes all print jobs for that user.**-** refers to

Table 33-5: Printer Commands

Printer Management	Description
	all print jobs for the logged-in user; if the logged-in user is the root, it refers to all print jobs.**-P***printer* removes all print jobs for the specified printer.
lpc	Manages your printers; at the **LPC** prompt, you can enter commands to check the status of your printers and take other actions.

Table 33-6: lpc Commands

Command	Operation
help *[command ...]*	Prints a short description of each command.
abort *printers*	Terminates an active spooling daemon on the local host immediately, and then disables printing for the specified printers; use «all» to indicate all printers.
clean *printers*	Removes any temporary files, data files, and control files that cannot be printed.
disable *printers*	Turns the specified printer queue off; new jobs are not accepted.
down *printers message*	Turns the specified printer queue off, disables printing, and puts *message* in the printer status file.
enable *printers*	Enables spooling for the listed printers; allows new jobs into the spool queue.
quit or **exit**	Exits from **lpc**.
restart *printers*	Starts a new printer daemon; used if the printer daemon, **lpd**, dies, leaving jobs yet to be printed.
reread	Read and execute the **lpd.conf** and **lpd.perms** configuration files.
start *printers*	Enables printing and starts a spooling daemon for the listed printers.
status *printers*	Displays the status of daemons and queues on the local machine.
stop *printers*	Stops a spooling daemon after the current job completes and disables printing.
topq *printer* [*jobnum* ...] [*user* ...]	Places the jobs in the order listed at the top of the printer queue.
up *printers*	Enables everything and starts a new printer daemon; undoes the effects of **down**.

Installing and Managing Printers with CUPS

The Common Unix Printing System (CUPS) is an alternative for LPRng that provides printing services. It is freely available under the GNU Public License. You can download a Red Hat version of CUPS from their Web site at **www.cups.org**. The site also provides detail documentation installing and managing printers. Whereas LPRng is derived from the old Berkeley line printer daemon (LPD), CUPS is based on the newer Internet Printing Protocol (IPP). The Internet Printing Protocol (**www.pwg.org/ipp**) is designed to establish a printing standard for the Internet. Whereas the older LPD-based printing systems focused primarily on line printers, an IPP-based system provides networking, postscript, and Web support. CUPS works like an Internet server and employs a configuration setup much like that of the Apache Web server. Its network support lets clients directly access printers on remote servers, without having to configure the printers themselves. Configuration needs only to be maintained on the print servers.

To install CUPS you first have to uninstall LPRng, along with printconf and printconf-gui. Use the **rpm –e** command to remove those applications. Bear in mind that you will lose your **/etc/printcap** file, so you may want to back up that file before you remove LPRng. An RPM package for CUPS is currently available on their Web site. For Red Hat 7.2 be sure to select the one for the 2.4 kernel or above. You can use the **rpm –i** command to install it.

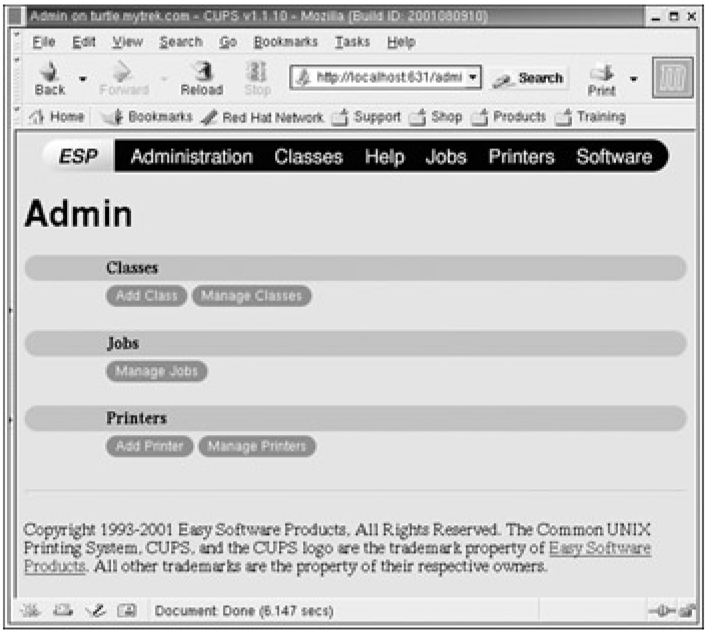
With the RPM version, a **cups** startup script is installed in the **/etc/rc.d/init.d** directory. You can start, stop, and restart CUPS using the service command and the **cups** script. When you make changes or install printers, be sure to restart CUPS to have your changes take effect.

```
service cups restart
```

The easiest way to configure and install printers with CUPS is to use its Web interface. This is a Web browser–based configuration tool like Webmin and SWAT. To start the Web interface, you open a browser like Mozilla and Netscape and enter the following URL:

```
http://localhost:631/admin
```

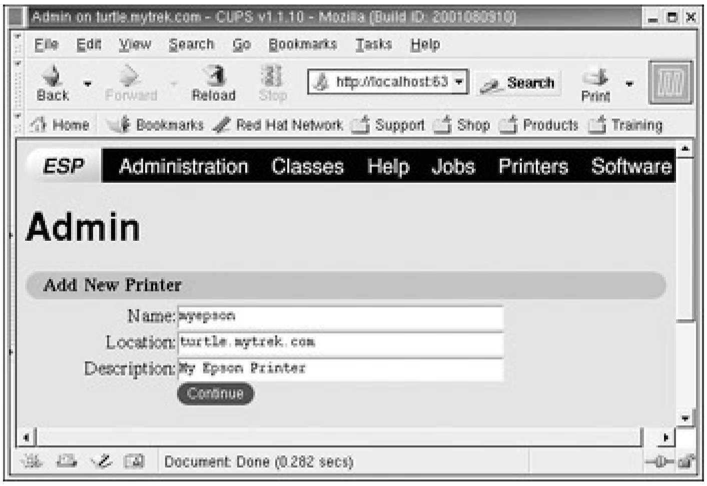
This displays the initial administration screen where you can manage print jobs and add printers (see Figure 33-5).



Copyright 1993-2001 Easy Software Products, All Rights Reserved. The Common UNIX Printing System, CUPS, and the CUPS logo are the trademark property of Easy Software Products. All other trademarks are the property of their respective owners.

Figure 33-5: CUPS Web interface

You install a printer on CUPS through a series of Web pages, each requesting different information. To install a printer, click on the Add Printer button to display a page where you enter the printer name and location. The location is the host to which the printer is connected. In Figure 33-6 the myepson printer is connected to the host **turtle.mytrek.com**.


Admin on turtle.mytrek.com - CUPS v1.1.10 - Mozilla (Build ID: 2001080910)
File Edit View Search Go Bookmarks Tasks Help
Back Forward Reload Stop
http://localhost:63
Search
Print
Home Bookmarks Red Hat Network Support Shop Products Training
ESP Administration Classes Help Jobs Printers Software
Continue

Admin

Add New Printer

Name: myepson

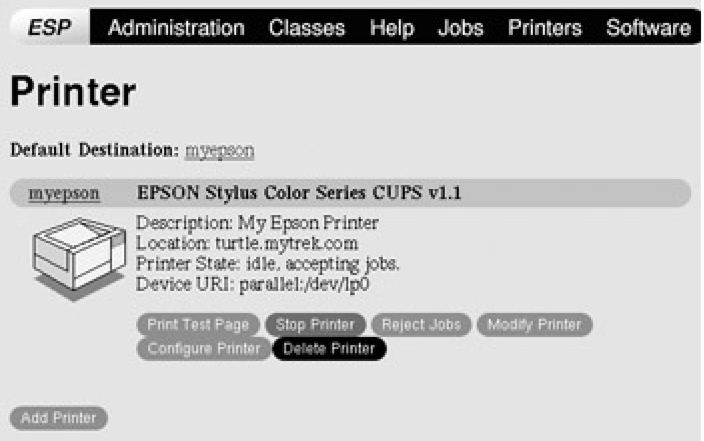
Location: turtle.mytrek.com

Description: My Epson Printer

Document: Done (0.282 secs)

Figure 33-6: CUPS Add New Printer page

Subsequent pages will prompt you to enter the model of the printer and driver. These you select from available listings. Once you have added the printer you can configure it. Clicking the Manage Printers entry in the Administration page will list your installed printers. You can then click a printer to display a page that will let you control the printer. You can stop the printer, configure its printing, modify its installation, and even delete the printer. Clicking the Configure Printer button will display a page where you can configure how your printer prints, specifying the resolution or paper size. The printer entry will be displayed as shown in Figure 33-7.



Printer

Default Destination: myepson

myepson **EPSON Stylus Color Series CUPS v1.1**

Description: My Epson Printer
Location: turtle.mytrek.com
Printer State: idle, accepting jobs.
Device URI: parallel:/dev/lp0

Figure 33-7: CUPS printer entry

Note You can perform all administrative tasks from the command line using the **lpadmin** command. See the CUPS documentation for more details.

To install a remote printer that is attached to a Windows system or another Linux system running LPRng or LPD, you specify its location using special URL protocols. For a Windows printer, you will first need to install, configure, and run Samba. CUPS uses Samba to access Windows printers. When installing the Windows printer on CUPS, you specify its location using the URL protocol **smb**. The user allowed to log on to the printer is entered before the host name and separated by the @ sign. On most configurations this is the guest user. The location entry for a Windows printer called myhp attached to a Windows host named lizard is shown here. Its Samba share reference would be **//lizard/myhp**.

```
smb://guest@lizard/myhp
```

To enable Samba on CUPS, you will also have to set the printing option in the **/etc/cups/cupsd.conf** file to Samba, as shown here.

```
printing = samba
```

You will also have to link the **smbspool** directory to the CUPS **smb** spool directory.

```
ln -s 'which smbspool`   /usr/cups/backend/smb
```

To access a printer connected to a Linux or Unix system running LPRng, you would use the protocol lpd when specifying its location. In the following example, the printer mylaser is connected to the Linux host rabbit.

```
lpd://rabbit.mytrek.com/mylaser
```

CUPS features a way to let you select a group of printers to print a job instead of selecting just one. That way, if one printer is busy or down, another printer can be automatically selected to perform the job. Such groupings of printers are called classes. Once you have installed your printers you can then group them into different classes. For example, you may want to group all inkjet printers in one class and lasers in another. Or you might want to group printers connected to a printer server in their own class. To create a class, select Classes on the Administration page, and enter the name of the class. You can then add printers to it.

CUPS configuration files will be placed in the **/etc/cups** directory. They are listed in Table 33-7. The **classes.conf**, **printers.conf**, and **client.conf** files can be managed by the Web interface. In the **printers.conf** file, you will see the configuration information for the different printers you have installed. Any of these files can be edited manually, if you wish.

Table 33-7: CUPS Configuration Files

Filename	**Description**
classes.conf	Configurations for different printer classes.
client.conf	Lists specific option for specified clients.
cupsd.conf	Configures the CUPS server, cupsd.
printers.conf	Printer configurations.

The CUPS server is configured with the **cupsd.conf** file. Configuration options have to be manually edited. The server is not configured with the Web interface. Your installation of CUPS will install a commented version of the **cupsd.conf** file, with each option listed, although mostly commented out. Commented lines are preceded with a # symbol and each option is documented in detail. The server configuration uses an Apache Web server syntax consisting of a set of directives. As with Apache, several of these directives can group other directives into blocks.

Certain directives allow access controls to be placed on specific locations. These can be printers or resources such as the administrative tool or the spool directories. Location controls are implemented with the Location directive. Allow From and Deny From directives can deny or permit access from specific hosts. CUPS supports both Basic and Digest forms of authentication, specified in the AuthType directive. Basic authentication uses a user and password. For example, to use the Web interface, you were prompted to enter the root user and the root user password. Digest authentication makes use of user and password information kept in the CUPS **/etc/cups/passwd.md5** file, using MD5 versions of a user and password for authentication. The AuthClass specifies the class allowed access. The System class include the root, sys, and system users. The following example shows the Location directive for the /admin resource, the administrative tool.

```
<Location /admin>

AuthType Basic
AuthClass System

## Restrict access to local domain
Order Deny,Allow
Deny From All
Allow From 127.0.0.1

#Encryption Required
</Location>
```

Installing and Managing Terminals and Modems

With a multiuser system such as Linux, you might have several users logged in at the same time. Each user would, of course, need his or her own terminal through which to access the Linux system. The monitor on your PC acts as a special terminal, called the *console,* but you can add other terminals either through the serial ports on your PC or a special multiport card installed on your PC. The other terminals can be stand-alone terminals or PCs using terminal emulation programs. For a detailed explanation of terminal installation, see the **Term-HOWTO** file in **/usr/doc/HOWTO**. A brief explanation is provided here.

The serial ports on your PC are referred to as COM1, COM2, on up to COM4. These serial ports correspond to the terminal devices **/dev/ttyS0** through **/dev/ttyS3**. Note, several of these serial devices may already be used for other input devices such as your mouse, and for communications devices such as your modem. If you have a serial printer, one of these serial devices is already used for that. If you installed a multiport card, you have many more ports from which to choose. For each terminal you add, you must create a character device on your Linux system. As with printers, you use the **mknod** command to create terminal devices. The permissions for a terminal device are 660. *Terminal devices* are character devices with a major number of 4 and minor numbers usually beginning at 64.

Terminal devices are managed by your system using the getty program and a set of configuration files. When your system starts, it reads a list of connected terminals in the **inittab** file and then executes an **/etc/getty** program for each one. The getty program sets up the communication between your Linux system and a specified terminal. It obtains from the **/etc/gettydefs** file certain parameters, such as speed and the login prompt, as well as any special instructions.

```
# Format: <speed># <init flags> # <final flags> #<login
 string>#<next-speed>
# 38400 fixed baud Dumb Terminal entry
DT38400# B38400 CS8 CLOCAL # B38400 SANE -ISTRIP CLOCAL #@S login:
 #DT38400
```

The **/etc/inittab** file holds instructions for your system on how to manage terminal devices. A line in the **/etc/inittab** file has four basic components: an ID, a runlevel, an action, and a process. Terminal devices are identified by ID numbers, beginning with 1 for the first device. The runlevel at which the terminal operates is usually 1. The action is usually *respawn,* which says to run the process continually. The process is a call to **/etc/getty** with the baud rate and terminal device name. The **/etc/ttys** file associates the type of terminal used with a certain device.

The **/etc/termcap** file holds the specifications for different terminal types. These are the different types of terminals users could use to log into your system. Your **/etc/termcap** file is already filled with specifications for most of the terminals currently produced. An entry in the **/etc/termcap** file consists of various names that can be used for a terminal separated by a **|** and then a series of parameter specifications, each ending in a colon. You find the name used for a specific terminal type here. You can use more to display your **/etc/termcap** file, and then use a search, /, to locate your terminal type. You can set many options for a terminal device. To change these options, use the **stty** command instead of changing configuration files directly. The **stty** command with no arguments lists the current setting of the terminal.

When a user logs in, having the terminal device initialized using the **tset** command is helpful. Usually the **tset** command is placed in the user's **.bash_profile** file and is automatically executed whenever the user logs into the system. You use the **tset** command to set the terminal type and any other options the terminal device requires. A common entry of **tset** for a **.bash_profile** file follows. The **-m dialup:** option prompts the user to enter a terminal type. The type specified here is a default type that is displayed in parentheses. The user presses ENTER to choose the default. The prompt looks like this: TERM=(vt100)?

```
eval 'tset -s -Q -m dialup:?vt00'
```

Input Devices

Input devices, such as mice and keyboards, are displayed on several levels. Initial configuration is performed during installation where you select the mouse and keyboard types. You can change that configuration with your administration configuration tool, such as Red Hat Setup (see Chapter 30). Red Hat provides a keyboard configuration tool called kbdconfig and a mouse configuration tool called mouseconfig. Both can be run from any command line interface. Special configurations also exist for mice and keyboard for the X Window System, and for the KDE and Gnome desktops. You select the keyboard layout and language, as well as configure the speed and display of the mouse.

Installing Sound, Network, and Other Cards

For you to install a new card, your kernel must be configured to support it. Support for most cards is provided in the form of modules that can be dynamically loaded in and attached to the kernel, running as its extension. Installing support for a card is usually a simple matter of loading a module that includes the drives for it. For example, drivers for the Sound Blaster sound card are in the module **sb.o**. Loading this module makes your sound card accessible to Linux. Most distributions automatically detect the cards installed on your system and load the needed modules. If you change cards, you may have to load the module you need manually, removing an older conflicting one. For example, if you change your Ethernet card, you may have to unload the module for your previous card and load in the one for your new card. Certain utilities, such as Linuxconf and netcfg**,** enable you to choose a new Ethernet card and have the module loaded for you. You can, however, load modules manually. The later "Modules" section in this chapter describes this process.

Device files for most cards are already set up for you in the **/dev** directory. For example, the device name for your sound card is **/dev/audio**. The device names for Network cards are aliases for network modules instead of device files. For example, the device name for your Ethernet card begins with **eth,** with the numbering starting from 0, as in **eth0** for the first Ethernet card on your system. They will alias the module used for that particular card; for example, a 3com Etherlink XL card will alias the 3c59x network module, whose alias would be **eth0** if it is the first Ethernet card.

Multimedia Devices: Sound, Video, and DVD

Currently, most Linux sound drivers are developed as part of the Open Sound System and freely distributed as OSS/Free. These are installed as part of Linux distributions. The OSS device drivers are intended to provide a uniform API for all Unix platforms, including Linux. They support Sound Blaster– and Windows Sound System–compatible sound cards (ISA and PCI). OSS is a commercial version called the *Open Sound System* (*OSS*). OSS is also available for a nominal fee and features configuration interfaces for device setup. A listing of the different OSS/Free sound devices is provided in Table 33-8. On Red Hat, you can use the sndconfig utility to install most sound cards on Linux. Some sound cards may require more specialized support (see Table 33-9). For sound cards, you can tell what your current sound configuration is by listing the contents of the **/dev/sndstat** file. You can test your card by simply redirecting a sound file to it, as shown here:

```
cat sample.au > /dev/audio.
```

Table 33-8: Sound Devices

Device	**Description**
/dev/sndstat	Sound driver status
/dev/audio	Audio output device
/dev/dsp	Sound sampling device
/dev/mixer	Control mixer on sound card
/dev/music	High-level sequencer
/dev/sequencer	Low-level sequencer
/dev/midi	Direct MIDI port

Table 33-9: Linux Sound Driver Sites

Driver Site	**Description**
Linux MIDI and Sound Pages	Information and links to Linux Sound projects and site: **www.xdt.com/ar/linux-snd**
Advanced Linux Sound	The Advanced Linux Sound Architecture project (ALSA) is

Table 33-9: Linux Sound Driver Sites

Driver Site	**Description**
Architecture (ALSA)	developed on Linux under the GPL: **www.alsa-project.org**
Open Sound System/Free	The standard Linux sound drivers formerly known as USS/Lite, TASD, and Voxware are now called OSS/Free: **www.opensound.com**
Open Sound System/Linux	OSS/Linux is a commercial version of the Linux sound drivers: **www.opensound.com**
Linux Ultra Sound Project	Drivers for the Gravis Ultrasound: **www.perex.cz/~perex/ultra**
PC Serial Port MIDI Driver	Linux MIDI driver for IBM-PC serial ports: **http://crystal.apana.org.au/ghansper/midiaxis.html**

The Linux Musical Instrument Digital Interface (MIDI) and Sound Pages currently at **www.xdt.com/ar/linux-snd** hold links to Web and FTP sites for Linux sound drivers for various sound cards. They also include links to sites for Linux MIDI and sound software.

The Advanced Linux Sound Architecture (ALSA) project is developing a modular sound driver, API, and configuration manager that aims to be a better alternative to OSS, while maintaining compatibility with it. ALSA is a GNU project and is entirely free; its Web site at **www.alsa-project.org** contains extensive documentation, applications, and drivers. Currently under development are the ALSA sound driver, the ALSA Kernel API, the ALSA library to support application development, and the ALSA manager to provide a configuration interface for the driver. ALSA evolved from the Linux Ultra Sound Project.

The Linux Ultra Sound Project has developed drivers for Gravix Ultrasound sound cards. Although Gravis Ultrasound is supported by OSS/Free, the Linux Ultra Sound Project drivers offer many more features. See Table 33-9 for a listing of sites providing sound drivers.

Table 33-10: Video Devices

Device Name	**Type of Device**
/dev/video	Video capture interface
/dev/vfx	Video effects interface
/dev/codec	Video codec interface
/dev/vout	Video output interface
/dev/radio	AM/FM radio devices
/dev/vtx	Teletext interface chips
/dev/vbi	Data services interface

Device names used for TV, video, and DVD devices are listed in Table 33-10. Drivers for DVD and TV decoders are also under development (see Table 33-11). mga4linux is developing video support for the Matrox Multimedia cards like the Marvel G200. The General ATI TV and Overlay Software (GATOS) is developing drivers for the currently unsupported features of ATI video cards, specifically TV features. The BTTV Driver Project has developed drivers for the Booktree video chip. Creative Labs sponsors Linux drivers for the Creative line of DVD DXR2 decoders (**opensource.creative.com**).

Table 33-11: Video, TV, and DVD Projects and Drivers

Device Drivers	**Type of Device**
linuxtv.org	Links to Video, TV, and DVD sites
video4linux 2	Video for Linux Two
LiViD	The Linux Video and DVD Project **www.linuxvideo.org**
LSDVD	LSDVD Linux Player Project **www.csh.rit.edu/projects/lsdvd/**
mga4linux	Driver for Matrox Multimedia Cards **www.cs.brandeis.edu/~eddie/mga4linux/**
GATOS	The General ATI TV and Overlay Software **www.core.binghamton.edu/~insomnia/gatos**
BTTV	BTTV drivers for cards with Booktree video chips **www.sourceforge.net**
DVD DXR2	Drivers for Creative DVD DXR2 decoders **opensource.creative.com**

Modules

Beginning with Linux kernel 2.0, the Linux kernel adopted a modular structure. In earlier kernel versions, support for specific features and devices had to be included directly into the kernel program. Adding support for a new device—say, a new kind of sound card—required you to create a new version of your kernel program that included the code for supporting that device. This involved a sometimes lengthy configuration, followed by compiling and installing the new kernel program, as well as making sure it was called properly when your system booted up.

As an alternative to this rebuilding of the kernel, Linux now supports the use of modules. *Modules* are components of the Linux kernel that can be loaded and attached to it as needed. To add support for a new device, you can now simply instruct a kernel to load its module. In some cases, you may have to recompile only that module to provide support for your device. The use of modules has the added advantage of reducing the size of the kernel program. The kernel can load modules in memory only as they are needed. For example, the module for the PPP network interface used for a modem only needs to be used when you connect to an ISP.

The modules your system needs are usually determined during installation, based on the kind of configuration information you provided. For example, if your system uses an Ethernet card whose type you specified during installation, the system loads the module for that card. You can, however, manually control what modules are to be loaded for your system. This, in effect, enables you to customize your kernel whatever way you want. You can use several commands, configuration tools, and daemons to manage kernel modules. The 2.4 Linux kernel includes the Kernel Module Loader (kmod), which has the capability to load modules automatically as they are needed. In addition, several tools enable you to load and unload modules manually, if you must. The Kernel Module Loader uses certain kernel commands to perform the task of loading or unloading modules. The **modprobe** command is a general-purpose command that calls **insmod** to load modules and **rmmod** to unload them. These commands are listed in Table 33-12. Options for particular modules, general configuration, and even specific module loading can be specified in the **/etc/modules.conf** file. You can use this file to automatically load and configure modules. You can also specify modules to be loaded at the boot prompt or in **lilo.conf** (see Chapter 3).

Table 33-12: Kernel Module Commands

Command	Description
lsmod	Lists modules currently loaded.
insmod	Loads a module into the kernel. Does not check for dependencies.
rmmod	Unloads a module currently loaded. Does not check for dependencies.
modinfo	Displays information about a module. **-a** (author), **-d** (description), **-p** (module parameters), **-f** (module filename), **-v** (module version).
depmod	Creates a dependency file listing all other modules on which the specified module may rely.
modprobe	Loads a module with any dependent modules it may also need. Uses the file of dependency listings generated by **depmod**. **-r** (unload a module), **-l** (list modules).

The filename for a module has the extension **.o**. Modules reside in the **/lib/modules/***version* directory, where *version* is the version number for your current module. The directory for the 2.4 kernel is **/lib/modules/2.4.7-10.** As you install new kernels on your system, new module directories are generated for them. One trick to access the directory for the current kernel is to use the **uname -r** command to generate the kernel version number. This command needs to have backquotes.

```
cd /lib/modules/'uname -r'
```

In this directory, modules for the kernel reside in the kernel directory. And within the kernel directory are several subdirectories, including the **drivers** directory that holds subdirectories for modules like the sound drivers or video drivers. These subdirectories serve to categorize your modules, making them easier to locate. For example, the **kernel/drivers/net** directory holds modules for your Ethernet cards, and the **kernel/drivers/sound** directory contains sound card modules.

Managing Modules with /etc/modules.conf

As noted previously, there are several commands you can use to manage modules. The **lsmod** command lists the modules currently loaded into your kernel, and **m**odinfo provides information about particular modules. Though you can use the **insmod** and **rmmod** commands to load or unload modules, you should only use **modprobe** for these tasks. See Table 33-12 for kernel module commands. It is often the case, however, that a given module requires other modules to be loaded. For example, the module for the Sound Blaster sound card, **sb.o**, requires the **sound.o** module to be loaded also. Instead of manually trying to determine what modules a given module depends on, you use the **depmod** command to detect the dependencies for you. The **depmod** command generates a file that lists all the modules on which a given module depends. The **depmod** command generates a hierarchical listing, noting what modules should be loaded first and in what order. Then, to load the module, you use the **modprobe** command using that file. **modprobe** reads the file generated by **depmod** and loads any dependent modules in the correct order, along with the module you want. You need to execute **depmod** with the **-a** option once, before you ever use **modprobe**. Entering **depmod -a** creates a complete listing of all module dependencies. This command creates a file called **modules.deb** in the module directory for your current kernel version, **/lib/modules/***version.*

```
depmod -a
```

To install a module manually, you use the **modprobe** command and the module name. You can add any parameters the module may require. The following command installs the Sound Blaster sound module with the I/O, IRQ, and DMA values. **modprobe** also supports the use of the * character to enable you to use a pattern to select several modules.

```
modprobe sb io=0x220 irq=5 dma=1
```

To discover what parameters a module takes, you can use the **modinfo** command with the **–p** option.

```
modinfo –p sb
```

You can use the **-l** option to list modules and the **-t** option to look for modules in a specified subdirectory. In the next example, the user lists all modules in the **sound** directory:

```
# modprobe -l -t sound
/lib/modules/2.4.7-10/kernel/drivers/sound/sb.o
/lib/modules/2.4.7-10/kernel/drivers/sound/sb_lib.o
/lib/modules/2.4.7-10/kernel/drivers/sound/sound.o
/lib/modules/2.4.7-10/kernel/drivers/sound/soundcore.o
```

Options for the **modprobe** command are placed in the **/etc/modules.conf** file. Here, you can enter configuration options, such as default directories and aliases. An alias provides a simple name for a module. For example, the following entry enables you to reference the 3c59x.o Ethernet card module as **eth0** (Kmod will automatically detect the 3Com Ethernet card and load the 3c59x module):

```
alias eth0 3c59x
```

Notice that there is no device name for Ethernet devices in the **/dev** directory. This is because the device name is really an alias for a Ethernet network module that has been defined in the **modules.conf** file. If you were to add another Ethernet card of the same type, you would place an alias for it in the **modules.conf** file. For a second Ethernet card, you would use the device name **eth1** as its alias. This way, the second Ethernet device can be referenced with the name **eth1**. A **modules.conf** entry is shown here:

```
alias eth1 3c59x
```

Note After making changes to **/etc/modules.conf**, you should run **depmod** again to record any changes in module dependencies.

The previous entry assumes that the Ethernet card was of the same model. If you had added a different model Ethernet card, you would have to specify the module used for that kind of card. In the following example, the second card is a standard PCI Realteck card. Kmod has already automatically detected the new card and loaded the ne2k-pci module for you. You only need to identify this as the **eth1** card in the **/etc/modules.conf** file.

```
alias eth0 3c59x
alias eth1 ne2k-pci
```

A sample **modules.conf** file is shown here. Notice the aliases for the USB controller and the sound card.

modules.conf

```
alias eth0 3c59x
alias eth1 ne2k-pci
alias parport_lowlevel parport_pc
alias usb-controller usb-uhci
alias sound-slot-0 i810_audio
```

Note In some cases, Kmod may not detect a device in the way you want, and thereby not load the kernel module you would like. This was the case in Chapter 32, where you needed to provide SCSI emulation for IDE CD write devices. In this case, entries in the **/etc/modules.conf** file were used to manually load modules, with certain options, overriding the original setup.

Installing New Modules for the Kernel

The source code for your Linux kernel contains an extensive set of modules, of which only a few are actually used on your system. When you install a new device, you may have to install the kernel module that provides the drivers for it. This involves selecting the module you need from a listing and then regenerating your kernel modules with the new module included. Then the new module is copied into the module library, installing it on your system. Then you can enter it in the **/modules.conf** file with any options, or use **modprobe** to install it manually.

First, make sure you have installed the kernel source code in the **/usr/src/linux***Version* directory (see Chapter 34). If not, simply use your distribution's installation utility such as rpm or an RPM utility like kpackage or Gnomerpm to install the kernel source RPM packages. The following command installs the kernel sources:

```
rpm -i kernel-source-2.4.7-10.i386.rpm
```

Now change to the **/usr/src/linux***Version* directory, where *Version* is the kernel version. Then use the **make** command with the **xconfig** or **menuconfig** argument to display the kernel configuration menus, invoking them with the following commands. The **make xconfig** command starts an X Window System interface that needs to be run on your desktop from a terminal window.

```
make xconfig
make menuconfig
```

Using the menus select the modules you need. Make sure each is marked as a module, clicking the Module check box in xconfig or pressing M for menuconfig. Once the kernel is configured, save it and exit from the configuration menus. Then you create the modules with the following command:

```
make modules
```

This places the modules in the kernel source modules directory: **/usr/src/linux***version***/**. You can copy the one you want to the kernel modules directory, **/lib/modules/***version***/kernel**, where *version* is the version number of your Linux kernel. A simpler approach is to reinstall all your modules, using the following command. This copies all the compiled modules to the **/lib/modules/***version***/kernel** directory.

```
make modules-install
```

For example, if you want to provide AppleTalk support and your distribution did not create an AppleTalk module or incorporate the support into the kernel directly, then you can use this method to create and install the AppleTalk modules. First, check to see if your distribution has already included it. The AppleTalk modules should be in the **/lib/modules/***version***/kernel/drivers/net/appletalk** directory. If not, you can move to the **/usr/src/linux***version* directory, run **make xconfig**, and select AppleTalk as a module. Then generate the modules with the **make modules** command. You could then use the **make modules-install** command to install the new module, along with your other modules. Or, you can copy the **appletalk** directory and the modules it holds to the module directory.

Chapter 34: Kernel Administration

Overview

The *kernel* is the core of the operating system, performing core tasks like managing memory and disk access, as well as interfacing with the hardware that makes up your system. For example, the kernel makes possible such standard Linux features as multitasking, which allows several users to work on the same system. It also handles communications with devices like your CD-ROM or hard disk. Users send requests for access to these devices through the kernel, which then handles the lower-level task of actually sending instructions to a device. Given the great variety of devices available, the system will vary in the kind of devices connect to a Linux system. These devices are automatically detected, and the kernel is appropriately configured when Linux is installed. However, if you add a new device, you may have to enable support for it in the kernel. This would involve creating a modified version of the kernel, often referred to as building or compiling the kernel. In addition, new versions of the kernel are continuously made available that will provide improved support for your devices, as well as a smoother running system. These you can easily download and install on your system. This chapter covers how you can download and install new kernels, as well as modify your current one.

The version number for a Linux kernel consists of three segments: the major, minor, and revision numbers. The *major number* increments with major changes in the kernel. The *minor number* indicates stability. *Even numbers* are used for stable releases, whereas *odd numbers* are reserved for development releases, which may be unstable. New features first appear in the development versions. If stability is a concern, waiting for the stable version is best. The *revision number* refers to the corrected versions. As bugs are discovered and corrected, new revisions of a kernel are released. A development kernel may have numerous revisions. For example, kernel 2.4.7 has a major number of 2 and a minor number of 4, with a revision number of 7. On Red Hat systems, another number is added that refers to a Red Hat–specific set of patches applied to the kernel. For Red Hat 7.2, this is 2.4.7-10, with 10 being the patch number. Currently, the newest kernel is 2.4.12, which also has a major number of 2 and a minor number of 4, but a revision number of 12. This is the most recent stable release of the Linux kernel. On Red Hat, which supports RPM packages, you can use an RPM query to learn what version is installed, as shown here:

```
rpm -q kernel
```

Note Unless you are experimenting with kernel development, you should always install a stable version of the kernel. The current stable version is 2.4, whereas 2.5 is the development version.

The Linux kernel is being worked on constantly, with new versions released when they are ready. Red Hat includes the most up-to-date kernel in its releases. Linux kernels are kept at **www.kernel.org**. Also, RPM packages for a new kernel often are available at distribution update sites, such as **ftp.redhat.com**. One reason you may need to upgrade your kernel is to provide support for new hardware or for features not supported by the distribution's version. For example, you may need support for a new device not provided in the distribution's version of the kernel. Certain features may not be included in a distribution's version because they are considered experimental or a security risk.

Note You probably don't need to install a new kernel only to add support for a new device. Kernels provide most device support in the form of modules, of which only those needed are installed with the kernel. Most likely your current kernel has the module you need. You simply have to install it. For this task, see the "Installing New Modules for the Kernel" section in Chapter 33.

You can learn more about the Linux kernel from **www.kernel.org**, the official repository for the current Linux kernels. The most current source code, as well as documentation, is here. For Red Hat systems, **www.redhat.com** also provides online documentation for installing and compiling the kernel on its systems. Several Linux HOW-TOs also exist on the subject. For Red Hat, consult the Red Hat Customization Guide for details on installing and compiling a kernel. The kernel source code software packages also include extensive documentation. Kernel source code files are always installed in the **/usr/src/linux** directory. In this directory, you can find a subdirectory named **Documentation,** which contains an extensive set of files and directories documenting kernel features, modules, and commands. See the following list of kernel resources.

Site	**Description**
www.kernel.org	The official Linux kernel Web site. All new kernels originate from here.
linuxhq.com	Linux headquarters, kernel sources, and patches.
ftp.redhat.com/pub/linux/updates/*version*/	Location of Red Hat packages for recent kernels, along with other updates. *version* is the number of your Red Hat distribution.

Precautionary Steps for Modifying Kernels

If you want to modify your current kernel, you should take care to retain your current one. Otherwise, your working kernel will be overwritten with the modified version. If something should go wrong, you would be unable to restore the previous working kernel. You should retain a copy of your current kernel so you can use it again in case something goes wrong with the new one. You do not have to worry about this happening if you are installing a new kernel. New kernels are given different names, so the older one is not overwritten.

To retain a copy of your current kernel, you can either make a backup copy of it, letting the original be overwritten, or modify the kernel source **Makefile** to have your modified version created with a different name. The kernel image file is called **vmlinuz-***version* where *version* is the version number attached, as in **vmlinuz-2.4.7. It** is located in the /**boot** directory. Also, there is a file called **/boot/vmlinuz**, which is only a symbolic link to the actual kernel file. When you generate a modified version of the kernel, the kernel file, here called **vmlinuz-2.4.7, will be overwritten with the new kernel image.**

To edit the **Makefile**, you change to the **/usr/src/linux-2.4** directory and carefully open the **Makefile** with a text editor. Then you locate the line that says EXTRAVERSION = and assign to it a unique name you want appended to the modified kernel to identify it. For example, if you appended a date, you could easily identify the kernel and know when you modified it.

```
EXTRAVERSION = -July2001
```

This would give you two kernels, keeping the original one. Corresponding map files and modules will also be generated. When you install your modified kernel, **/boot/vmlinuz** will link to it, in this case, **vmlinuz-2.4.7-July2001**.

```
/boot/vmlinuz-2.4.7
/boot/vmlinuz-2.4.7-July2001
```

Making a backup copy is a more intuitive process, but takes a few more steps. You would make a copy of the **/boot/mvlinux-2.4.7** file, giving it another name as shown here:

```
cp /boot/vmlinuz2.4.7-10 /boot/vmlinuz2.4.7-10.back
```

You could also make a backup of the **System.map** file. This file contains kernel symbols needed by modules to start kernel functions. In the case of kernel 2.4.7-10, this would be **System.map-2.4.7-10**. You should also back up your modules located in the **/lib/modules/***version* directory, where *version* is the version number of the kernel. Otherwise, you will lose the modules already set up to work with the original kernel. For version 2.4.7-10, the libraries are located in **/lib/modules/2.4.7-10**. If you are compiling a different version, those libraries are placed in a new directory named with the new version number.

If you are using a boot loader, you should create a new entry for the old kernel in the boot loader configuration file (see Chapter 29). You can then make an entry for the new kernel in that configuration file. Leaving the entry for the old kernel is advisable in case something goes wrong with the new kernel. This way, you can always reboot and select the old kernel. For example, in the **grub.conf** file, add a new entry, similar to the one for the old kernel, which references the new kernel in its image line. The **grub.conf** entry would look something like the following code. You could then select the entry with the title Old Red Hat Linux (2.4.7-10.back) from the GRUB menu to launch the old kernel.

```
title Old Red Hat Linux (2.4.7-10.back)
       root (hd0,2)
       kernel /boot/vmlinuz-2.4.7-10.back ro root=/dev/hda3
       initrd /boot/initrd-2.4.7-10.back.img
```

Also advisable is to have a boot disk ready, just in case something goes wrong with the installation. With a boot disk, you can start your system without using the boot loader. On Red Hat systems, you can create a boot disk using the **mkbootdisk** utility. To create a boot disk, you will need to know the full version number for your kernel. You can, in fact, have several kernels installed and create boot disks for each one (your **grub.conf** will list your kernel version number). For Red Hat 7.2, the kernel version is 2.4.7-10. Use it as the argument to the **mkbootdisk** command to create the bootdisk for your system:

```
mkbootdisk 2.4.7-10
```

Installing Distribution Kernel Binaries and Source: RPM

To install a new kernel on Red Hat, you need to download the software packages for that kernel to your system. It is advisable to download the RPM packages for new kernels from the Red Hat FTP site. Alternatively, you can download the most recent version from **www.kernel.org**. You can install a new kernel either by downloading a binary version from your distribution's Web site and installing it or by downloading the source code, compiling the kernel, and then installing the resulting binary file. For Red Hat, the binary version of the kernel is provided in an RPM package. You can install a new kernel using the Red Hat Package Manager, just as you would any other RPM software package.

The easiest way to install a new kernel on Red Hat is to use the Red Hat Update Agent. The Agent will not automatically select kernel file for download. Though listed, you have to explicitly select them for them to be downloaded and installed.

The source code version is available either from the Red Hat FPT site, **ftp.redhat.com** for Red Hat, or from **www.kernel.org**. Wherever you download a kernel version from, it is always the same. The source code downloaded for a particular kernel version from Red Hat is the same as the one for **www.kernel.org**. Patches for that version can be applied to any distribution.

If you wish to download kernel RPM packages directly from **ftp://ftp.redhat.com**, you will need to locate the corresponding **updates** directory for your Red Hat distribution, such as 7.1. A series of RPM packages are there, all beginning with the term *kernel.* There are also other packages you may need, which contain updated system configuration files used by the new kernel. As an example, the kernel packages for the 2.4 kernel are listed in the following code. Only install one of the **kernel-*version*-ix86** packages and one of the **kernel-smp** packages. Choose the one for your machine—for example, i686 for a Pentium II, i586 for a Pentium, and i386 for other PCs. The following list shows the RPM packages for the Red Hat version of kernel 2.4:

```
kernel-2.4.7-10.i386.rpm
kernel-2.4.7-10.althion.rpm
kernel-2.4.7-10.i686.rpm
```

```
kernel-doc-2.4.7-10.i386.rpm
kernel-enterprise-2.4.7-10.i686.rpm
kernel-headers-2.4.7-10.i386.rpm
kernel-pcmcia-cs-2.4.7-10.i386.rpm
kernel-smp-2.4.7-10.i386.rpm
kernel-smp-2.4.7-10.althion.rpm
kernel-smp-2.4.7-10.i686.rpm
kernel-source-2.4.7-10.i386.rpm
```

To make sure a kernel RPM package was downloaded without any errors, you can use the **rpm** command with the **-K --nopgp** options to check it:

```
rpm -K --nopgp *rpm
```

You are now ready to install the new kernel. First, install updated versions, if any, of other support packages. In Red Hat, these currently include mkinitrd, SysVinit, and initscripts. Use the **-Uvh** option to update those packages:

```
# rpm -Uvh mkinitrd*rpm SysVinit*rpm initscripts*rpm
```

Installing the source code and headers for the kernel is also essential. You use the source code to generate any modules and tailor the kernel to your own needs. For example, you can use the source code to generate modules containing devices drivers for any uncommon devices you may have installed, as shown here:

```
# rpm -Uvh kernel-headers-2.4.7-10.i386.rpm
# rpm -Uvh kernel-source-2.4.7-10.i386.rpm
```

You can now install the kernel. On Red Hat systems, you install the kernel, kernel-ibcs, and the kernel-pcmia-cs packages. As a safety precaution, it is advisable to preserve your old kernel in case the new one does not work out for some reason. This involves installing with the install (**-i**) option instead of the update(**-U**) option, creating a separate RAM disk for the new kernel, and then modifying **grub.conf** to have GRUB start up using the new kernel.

```
# rpm -Uvh kernel-2.4.7-10.i686.rpm
# rpm -Uvh kernel-pcmcia-cs-2.4.7-10.386.rpm
# rpm –Uvh kernel-smp-2.4.7-10.i686.rpm
```

On Red Hat, kernels are installed in the **/boot** directory. Performing an **ls -l** operation on this directory lists all the currently installed kernels. A file for your old kernel and a file for your new one now exist, as well as a link file called **vmlinuz** that links to the new kernel file. If you took the precautions described in the previous section, you may have already renamed the older kernel. On Red Hat, if you are using a boot loader like GRUB, you needn't change its configuration file (grub.conf) because the entry to invoke the kernel still references the **/boot/vmlinuz** link, which now points to the new kernel. Red Hat boots the kernel using the **/boot/vmlinux** link to the kernel file. In your **grub.conf** file, the kernel line for the kernel file references this link.

```
 kernel = /boot/vmlinuz-2.4.7-10
```

Compiling the Kernel from Source Code

Instead of installing already compiled binary versions of the kernel for Red Hat, you can install the kernel source code on your system and use it to create the kernel binary files yourself. Kernel source code files are compiled with the gcc compiler just as any other source code files are. One advantage to compiling the kernel is you are able to customize its configuration, selecting particular devices you want supported by the kernel or the kind of networking support you want. You can have more control over exactly what your operating system can support. The 2.4 kernel is described here.

Installing Kernel Sources: Kernel Archives and Patches

You can obtain a recent version of the kernel source code from Red Hat. It will have the name **kernel-source**. The **kernel-source** file is usually installed as part of your installation. New versions can be downloaded with the Red Hat Update Agent, or by directly accessing the Red Hat FTP site. Be sure to download both the kernel headers and the source code RPM files. As noted previously, you simply install them as you would any RPM package.

```
# rpm -Uvh kernel-headers-2.4.7-10.i386.rpm
# rpm -Uvh kernel-source-2.4.7-10.i386.rpm
```

The source files are placed in the **/usr/src** directory, within the subdirectory that will have the prefix *linux* and a suffix consisting of the kernel version, as in **linux-2.4.7** for kernel 2.4, release 2, patch 7. The full directory will be **/usr/src/linux-2.4.7**. When you download and install a new kernel, a separate subdirectory will be created for it. For example, the 2.4.7-10 kernel will be placed in **/usr/src/linux-2.4.3**. A link is created called **/usr/src/linux-2.4** that will link to the most recent kernel source directory that you installed. You can use this link to access your most recent kernel source. Originally, this would link to **/usr/src/linux-2.4.7**. If you later installed the 2.4.7-10 kernel, this would link to **/usr/src/linux2.4.3**.

You can also obtain the most recent version of the source code from www.kernel.org. These versions are normally much more recent that those available on Red Hat, but may not have been thoroughly tested on the Red Hat platform. The kernel source will be in the form of compressed archives (**.tar.gz**). They will have the prefix **linux** with the version name as the suffix. For example, **linux-2.4.6** is the 2.4 kernel, revision 6. You first decompress and extract the archive with the following commands. *vnum* is the version number. First you change to the **/usr/src** directory and then unpack the archive. It will create a directory called **linux** where the source files are placed. The following example extracts the 2.4.6 kernel:

```
cd /usr/src
gzip -cd linux-2.4.6.tar.gz | tar xvf -
```

Be sure to unpack it in the **/usr/src** directory. The archive extracts a directory named **linux** that holds the source code files. This way, the files are located in the **/usr/src/ linux** directory.

Once you have extracted your kernel source, you should download and apply any patches. A *patch* modifies a source code file, making required changes. To install a patch, download the patch file and then execute the following command. The patch file is first decompressed, and then the **patch** command implements the changes.

```
cd /usr/src
```

```
gzip -cd patchvnum.gz | patch -p0
```

You can also decompress the patch and then redirect it to the **patch** command. **patch** reads its patches from the standard input.

```
gunzip patch-2.4.6.gz
patch -p0 < patch-2.4.6
```

You may have to implement several patches, depending on how out-of-date your kernel is. In this case, you must execute a patch operation for each patch file needed. Patches need to be applied in sequence. The latest patch does not include any previous ones. So to apply patch 2.4.7, you would first have to apply patch 2.4.1. Or, you can use the patch-kernel script, which determines your kernel version and applies the patches needed.

```
cd /usr/src
linux/scripts/patch-kernel linux
```

Before you can install the kernel, you have to configure and compile it, as discussed in the next section.

Note Once you have installed a kernel source for a particular revision, you can update it by downloading and installing any patches for it.

Configuring the Kernel

Once the source is installed, you must configure the kernel. Configuration consists of determining the features for which you want to provide kernel-level support. This includes drivers for different devices, such as sound cards and SCSI devices. This process is referred to as *configuring the kernel*. You can configure features as directly included in the kernel itself or as modules the kernel can load as needed. You can also specifically exclude features. Features incorporated directly into the kernel make for a larger kernel program. Features set up as separate modules can also be easily updated. Documentation for many devices that provide sound, video, or network support can be found in the **/usr/share/doc** directory. Check the kernel-doc package to find a listing of the documentation provided.

```
rpm -ql kernel-doc
```

Note If you configured your kernel previously and now want to start over from the default settings, you can use the **make mrproper** command to restore the default kernel configuration.

You can configure the kernel using one of several available configuration tools: config, menuconfig, and xconfig. They perform the same configuration tasks, but use different interfaces. The config tool is a simple configure script providing line-based prompts for different configuration options. The menuconfig tool provides a cursor- based menu, which you can still run from the command line. Menu entries exist for different configuration categories, and you can pick and choose the ones you want. To mark a feature for inclusion in the kernel, move to it and press the SPACEBAR. An asterisk appears in the empty parentheses to the left of the entry. If you want to make it a module, press M and an *M* appears in the parentheses. The xconfig tool runs on a window manager and provides a window interface with buttons and menus. You can use your mouse to select entries. A menu consists of configuration categories that are listed as buttons you can click. All these tools save their settings to the **.config** file in the kernel source's directory. Should you want to remove a configuration entirely, you can use the **mrproper** option to remove the **.config** file, starting over from scratch.

```
make mrproper
```

You start a configuration tool by preceding it with the **make** command. Be sure you are in the **/usr/src/linux-***version* directory. The process of starting a configuration tool is a **make** operation that uses the Linux kernel **Makefile**. The xconfig tool should be started from a terminal window on your window manager. The menuconfig and config tools are started on a shell command line. The following example lists commands to start xconfig, menuconfig, or config:

```
make xconfig
make menuconfig
make config
```

The xconfig tool opens a Linux Kernel Configuration window listing the different configuration categories. Figure 34-1 shows the configuration categories for the 2.4 kernel. Buttons at the right of the screen are used to save the configuration or to copy it to a file, as well as to quit. Clicking an entry opens a window that lists different features you can include. Three check boxes to the left of each entry enable you to choose to have a feature compiled directly into the kernel, created as a separate module that can be loaded at runtime, or not included at all. As a rule, features in continual use, such as network and file system support, should be compiled directly into the kernel. Features that could easily change, such as sound cards, or features used less frequently, should be compiled as modules. Otherwise, your kernel image file may become too large and slower to run.

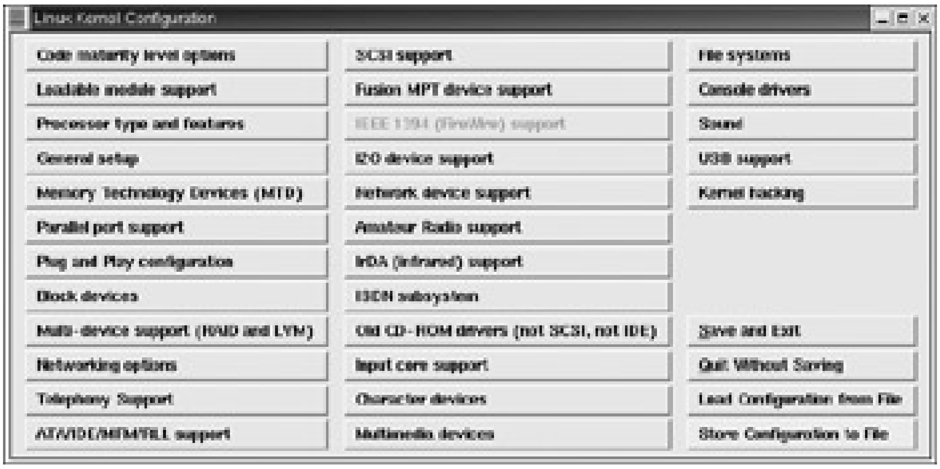


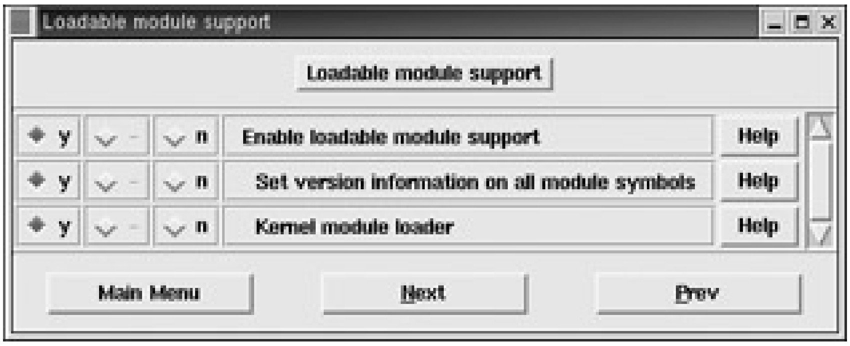
Figure 34-1: The xconfig Linux kernel configuration tool

Note If you decide to include a feature directly into the kernel that was previously a module, be sure to check that the old module is removed from the **/lib/modules/**version directory. Otherwise, conflicts can occur between the module and its corresponding code, which is now directly part of the kernel.

The xconfig and menuconfig tools provide excellent context-sensitive help for each entry. To the right of an entry is a Help button. Click it to display a detailed explanation of what that feature does and why you would include it either directly or as a module, or even exclude it.

When in doubt about a feature, always use the Help button to learn exactly what it does and why you would want to use it. Many of the key features are described here.

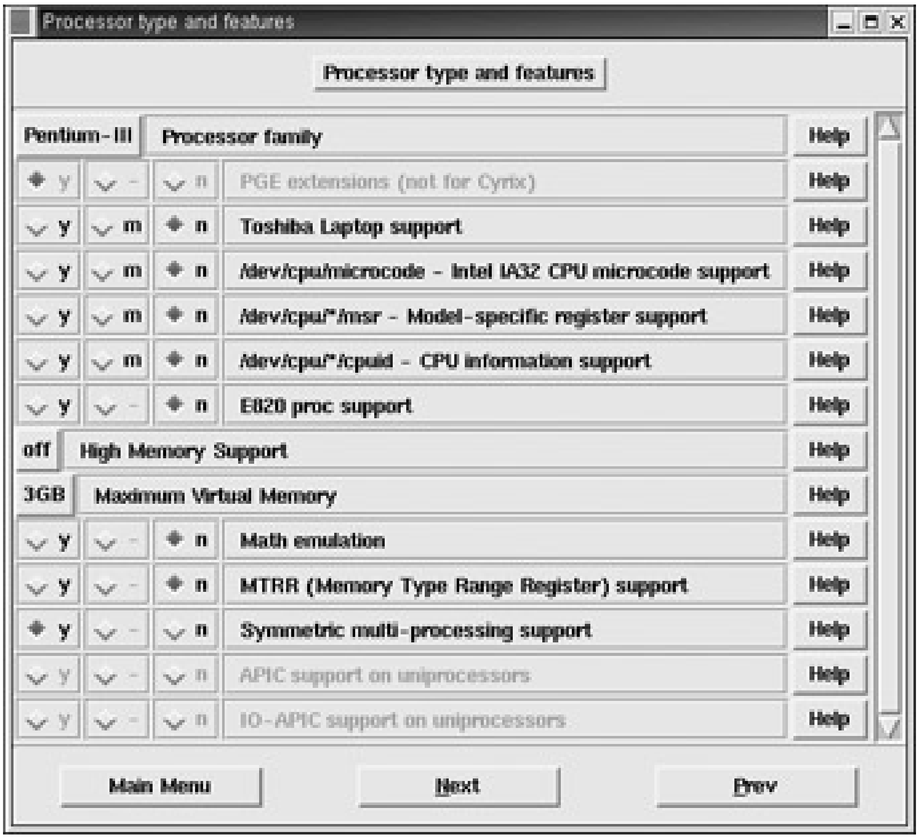
- **Loadable Module Support** In most cases, you should make sure your kernel can load modules. Click the Loadable Modules Support button to display a listing of several module management options (see Figure 34-2). Make sure Enable Loadable Module Support is marked Yes. This feature allows your kernel to load modules as they are needed. Kernel Module Loader should also be set to Yes, as this allows your daemons, like your Web server, to load any modules they may need. The Set Version Information entry enables you to use any modules set up for previous kernels.



y	-	n	Option	
◆ y	◇ -	◇ n	Enable loadable module support	Help
◆ y	◇ -	◇ n	Set version information on all module symbols	Help
◆ y	◇ -	◇ n	Kernel module loader	Help

Figure 34-2: Loadable modules support

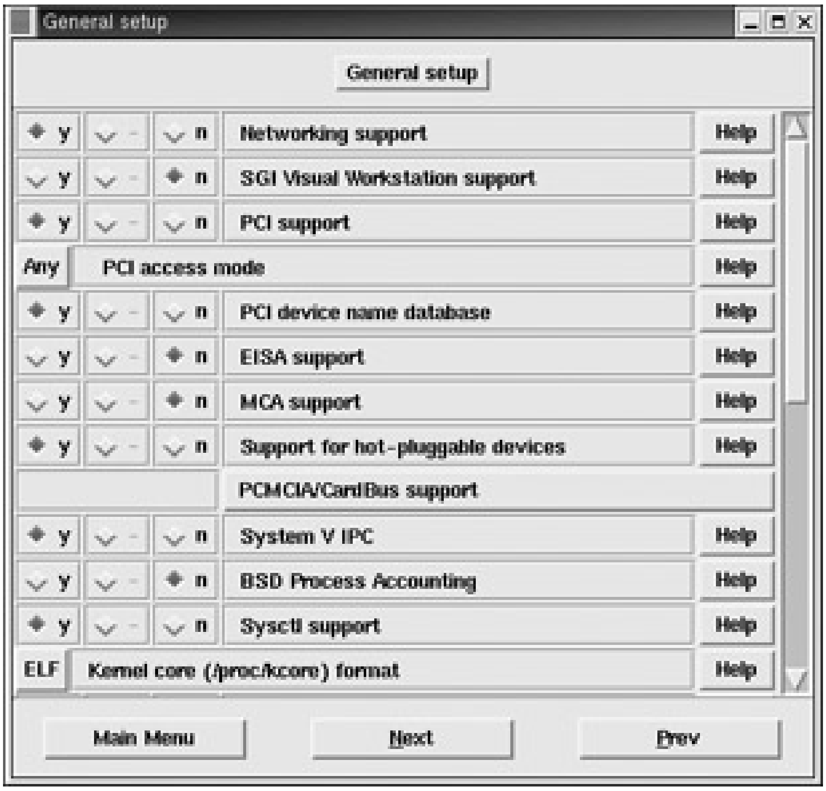
- **Processor Type and Features** The Processor Type and Features window enables you to set up support for your particular system (see Figure 34-3). Here, you select the type of processor your have (486, 586, 686, Pentium III, and so forth), as well as the amount of maximum memory your system supports (up to 64 gigs with 2.4 kernel).



y	m	n	Option	
Pentium-III			Processor family	Help
◆ y	◇ -	◇ n	PGE extensions (not for Cyrix)	Help
◇ y	◇ m	◆ n	Toshiba Laptop support	Help
◇ y	◇ m	◆ n	/dev/cpu/microcode - Intel IA32 CPU microcode support	Help
◇ y	◇ m	◆ n	/dev/cpu/*/msr - Model-specific register support	Help
◇ y	◇ m	◆ n	/dev/cpu/*/cpuid - CPU information support	Help
◇ y	◇ -	◆ n	E820 proc support	Help
off			High Memory Support	Help
3GB			Maximum Virtual Memory	Help
◇ y	◇ -	◆ n	Math emulation	Help
◇ y	◇ -	◆ n	MTRR (Memory Type Range Register) support	Help
◆ y	◇ -	◇ n	Symmetric multi-processing support	Help
◇ y	◇ -	◇ n	APIC support on uniprocessors	Help
◇ y	◇ -	◇ n	IO-APIC support on uniprocessors	Help

Figure 34-3: Processor Type and Features window

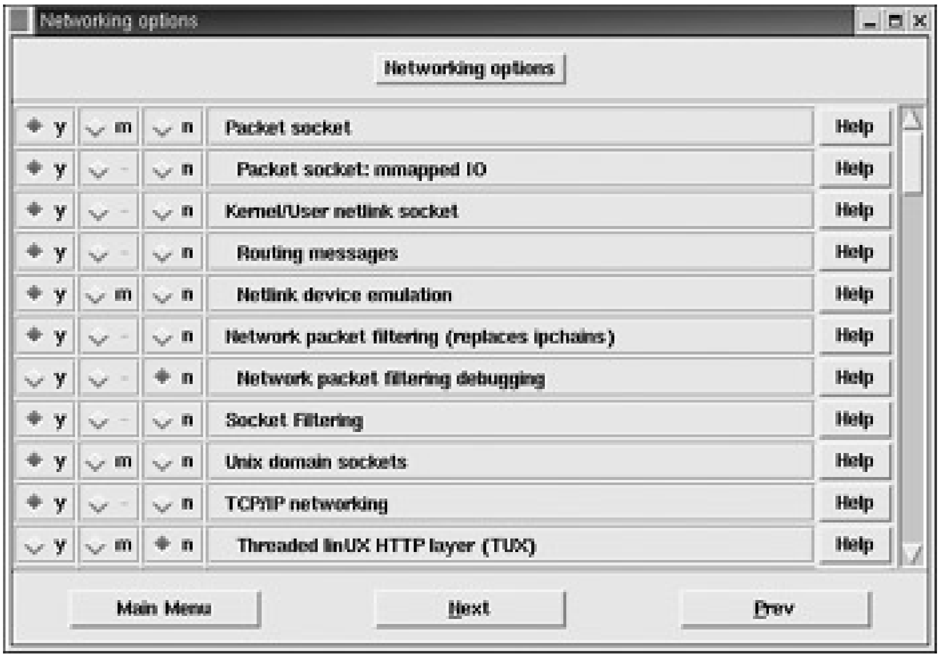
- **General Setup** The General Setup window enables you to select general features, such as networking, PCI BIOS support, and power management, as well as support for ELF and a.out binaries (see Figure 34-4).



● y	○ -	○ n	Networking support	Help
○ y	○ -	● n	SGI Visual Workstation support	Help
● y	○ -	○ n	PCI support	Help
Any			PCI access mode	Help
● y	○ -	○ n	PCI device name database	Help
○ y	○ -	● n	EISA support	Help
○ y	○ -	● n	MCA support	Help
● y	○ -	○ n	Support for hot-pluggable devices	Help
			PCMCIA/CardBus support	
● y	○ -	○ n	System V IPC	Help
○ y	○ -	● n	BSD Process Accounting	Help
● y	○ -	○ n	Sysctl support	Help
ELF			Kernel core (/proc/kcore) format	Help

Figure 34-4: General Setup window

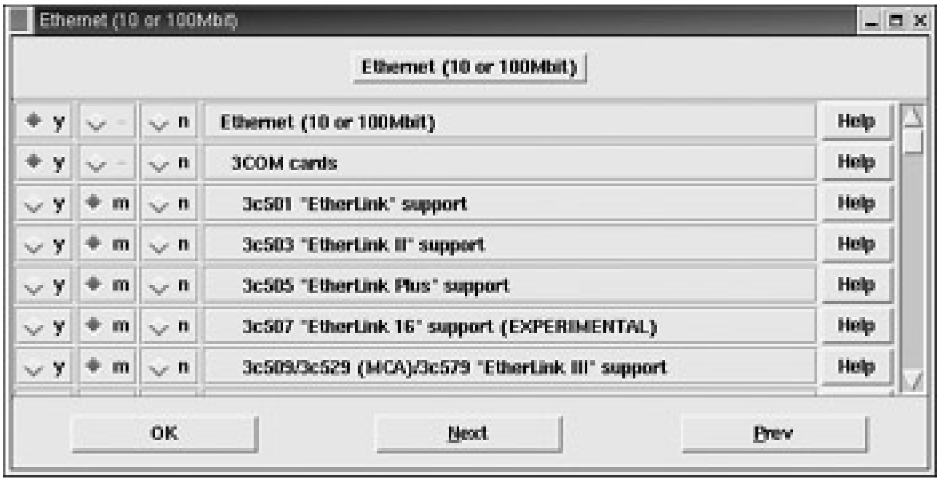
- **Block Devices** The Block Devices window lists entries that enable support for your IDE, floppy drive, and parallel port devicesRAD. Special features, such as RAM disk support and the loopback device for mounting CD-ROM image files, are also there.
- **Multi-Device Support** The Multi-Device Support window lists entries to enable the use of RAID devices. You can choose the level of RAID support you want.
- **Networking Options** The Networking Options window, shown in Figure 34-5, lists an extensive set of networking capabilities. The TCP/IP Networking entry must be set to enable any kind of Internet networking. Here, you can specify features that enable your system to operate as a gateway, firewall, or router. Network Aliasing enables support for IP aliases. Support also exists for other kinds of networks, including AppleTalk and IPX. AppleTalk must be enabled if you want to use NetaTalk to connect to a Macintosh system on your network.



● y	○ m	○ n	Packet socket	Help
● y	○ -	○ n	Packet socket: mmapped IO	Help
● y	○ -	○ n	Kernel/User netlink socket	Help
● y	○ -	○ n	Routing messages	Help
● y	○ m	○ n	Netlink device emulation	Help
● y	○ -	○ n	Network packet filtering (replaces ipchains)	Help
○ y	○ -	● n	Network packet filtering debugging	Help
● y	○ -	○ n	Socket Filtering	Help
● y	○ m	○ n	Unix domain sockets	Help
● y	○ -	○ n	TCP/IP networking	Help
○ y	○ m	● n	Threaded linUX HTTP layer (TUX)	Help

Figure 34-5: Networking Options window

- **ATA/IDE/MFM/RLL Support** In the ATA/IDE/MFM/RLL Support window, you can click on the IDE, ATA, and ATAPI Block Device button to open a window where you can select support for IDE ATA hard drives and ATAPI CD-ROMs. Included here are IDE chipsets such as HTP366 used for ATA66 drives.
- **SCSI Support** If you have any SCSI devices on your system, make sure the entries in the SCSI Support window are set to Yes. You enable support for SCSI disks, tape drives, and CD-ROMs here. The SCSI Low-Level Drivers window displays an extensive list of SCSI devices currently supported by Linux. Be sure the ones you have are selected.
- **Network Device Support** The Network Device Support window lists several general features for network device support. There are entries here for windows that list support for particular types of network devices, including Ethernet (10 or 100Mbit) devices, token ring devices, WAN interfaces, and AppleTalk devices. Many of these devices are created as modules you can load as needed. You can elect to rebuild your kernel with support for any of these devices built directly into the kernel. Figure 34-6 shows the Ethernet (10 or 100Mbit) window listing Ethernet devices. Notice they are built as separate modules.

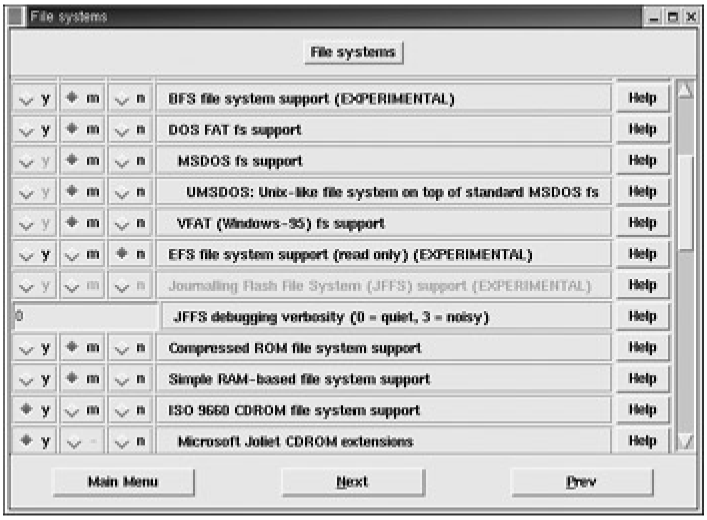


Ethernet (10 or 100Mbit)

● y	○ -	○ n	Ethernet (10 or 100Mbit)	Help
● y	○ -	○ n	3COM cards	Help
○ y	● m	○ n	3c501 "EtherLink" support	Help
○ y	● m	○ n	3c503 "EtherLink II" support	Help
○ y	● m	○ n	3c505 "EtherLink Plus" support	Help
○ y	● m	○ n	3c507 "EtherLink 16" support (EXPERIMENTAL)	Help
○ y	● m	○ n	3c509/3c529 (MCA)/3c579 "EtherLink III" support	Help

Figure 34-6: Ethernet (10 or 100Mbit) window

- **Multimedia Devices**   Multimedia devices provide support for various multimedia cards as well as Video4Linux.
- **File Systems**   The File Systems window, shown in Figure 34-7, lists the different types of file systems Linux can support. These include DOS, VFAT (Windows 95), and ISO9660 (CD-ROM) file systems. Network file systems— such as NFS, SMB (Samba), NCP (NetWare), HFS (Macintosh), and NTFS—are also listed. Note, the Linux file system type, **ext2fs**, must be included in the kernel, and is not compiled as a module.

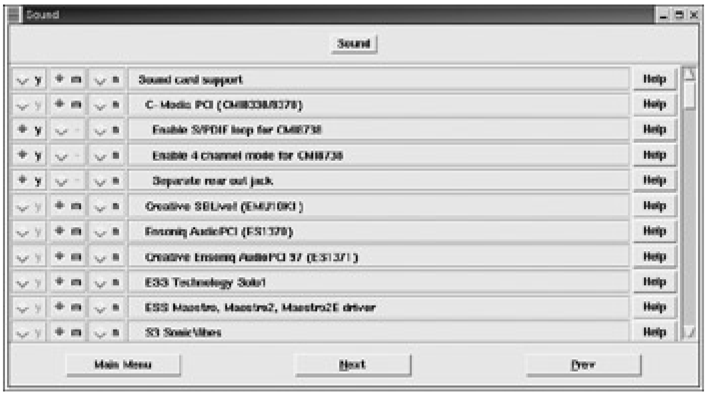


File systems

○ y	● m	○ n	BFS file system support (EXPERIMENTAL)	Help
○ y	● m	○ n	DOS FAT fs support	Help
○ y	● m	○ n	MSDOS fs support	Help
○ y	● m	○ n	UMSDOS: Unix-like file system on top of standard MSDOS fs	Help
○ y	● m	○ n	VFAT (Windows-95) fs support	Help
○ y	○ m	● n	EFS file system support (read only) (EXPERIMENTAL)	Help
○ y	○ m	○ n	Journalling Flash File System (JFFS) support (EXPERIMENTAL)	Help
0			JFFS debugging verbosity (0 = quiet, 3 = noisy)	Help
○ y	● m	○ n	Compressed ROM file system support	Help
○ y	● m	○ n	Simple RAM-based file system support	Help
● y	○ m	○ n	ISO 9660 CDROM file system support	Help
● y	○ -	○ n	Microsoft Joliet CDROM extensions	Help

Figure 34-7: File Systems window

- **Character Devices**   The Character Devices window lists features for devices such as your keyboard, mouse, and serial ports. Support exists for both serial and bus mice.
- **Sound**   The Sound window lists different sound cards supported by the kernel. Select the one on your system. You also must provide the IRQ, DMA, and Base I/0 your sound card uses. These are compiled as separate modules, some of which you could elect to include directly in the kernel if you want (see Figure 34-8).



○ y	● m	○ n	Sound card support	Help
○ y	● m	○ n	C-Media PCI (CMI8338/8378)	Help
● y	○ -	○ n	Enable S/PDIF loop for CMI8738	Help
● y	○ -	○ n	Enable 4 channel mode for CMI8738	Help
● y	○ -	○ n	Separate rear out jack	Help
○ y	● m	○ n	Creative SBLive! (EMU10K1)	Help
○ y	● m	○ n	Ensoniq AudioPCI (ES1370)	Help
○ y	● m	○ n	Creative Ensoniq AudioPCI 97 (ES1371)	Help
○ y	● m	○ n	ESS Technology Solo1	Help
○ y	● m	○ n	ESS Maestro, Maestro2, Maestro2E driver	Help
○ y	● m	○ n	S3 SonicVibes	Help

Figure 34-8: Sound window

- **Kernel Hacking** The Kernel Hacking window lists features of interest to programmers who want to modify the kernel code. You can have the kernel include debugging information.

Once you set your options, save your configuration. You can also make a backup copy by clicking Save To File.

Compiling and Installing the Kernel

Now that the configuration is ready, you can compile your kernel. You first need to generate a dependency tree to determine what part of the source code to compile, based on your configuration. Use the following command:

```
make dep
```

You also have to clean up any object and dependency files that may remain from a previous compilation. Use the following command to remove such files:

```
make clean
```

You can use several options to compile the kernel (See Table 34-1 later in this chapter). The **bzImage** option simply generates a kernel file called **bzImage** and places it in the **arch** directory. For Intel systems, you find **bzImage** in the **i386/boot** subdirectory, **arch/i386/boot**. For a Red Hat kernel source, this would be in **/usr/src/linux-2.4 /arch/i386/boot**. For a kernel archive source, this would be in **/usr/src/linux/arch /i386/boot**.

```
make bzImage
```

The **install** option generates both the kernel files and installs them on your system, as **vmlinuz**.

```
make install
```

The **zlilo** option installs the kernel file as well, but also runs **lilo** to update LILO. The **zlilo** option is designed for use with systems that run LILO. If you are booting Linux from DOS using **loadlin**, you will need to copy the **bzImage** file to the **loadlin** directory on the DOS partition where you are starting Linux from.

To install a kernel **bzImage** file manually, copy the **bzImage** file to the directory where the kernel resides and give it the name used on your distribution, such as **vmlinuz** for Red Hat. Remember to first back up the old kernel file. On Red Hat, **vmlinuz** is a symbolic link to an actual kernel file that will have the term "vmlinuz" with the version name. So, to manually install a **bzImage** file on Red Hat, you copy it to the **/boot** directory with the name **vmlinuz** and the attached version number such as **vmlinuz-2.4.7-10**. You then create a symbolic link from **/boot/vmlinuz** to **/boot/vmlinuz-2.4.7-10**.

```
make bzImage
cp arch/i386/boot/bzImage /boot/vmlinuz-2.4.7-10
ln -s /boot/vmlinuz /boot/vmlinuz-2.4.7-10
```

The **bzImage** option and those options that begin with the letter *b*, like **bzlilo**, create a compressed kernel image. This kernel image may not work on older systems. If not, try using the **zImage** option to create a kernel file called **zImage**. Then install the **zImage** file manually. Bear in mind that support for **zImage** will be phased out eventually.

```
make zImage
```

The **bzlilo** option both installs the kernel file on your system and runs LILO. You should use this option if you receive an error saying your kernel is too large. The following command compiles the kernel, installs it on your system, and runs LILO for you. To install and update LILO, use the following:

```
make zlilo
```

If you receive an error saying the kernel is too large, try using a *b* version of the option, such as **bzlilo**, to further reduce its size.

```
make bzlilo
```

If you want to create a boot disk with the new kernel, use the **bzdisk** option. This option will install the kernel on a floppy disk placed in your floppy drive. The kernel will reside on the floppy disk, and to use that kernel you boot your system from the floppy (the kernel is not installed on your root partition as it is with the install option).

```
make bzdisk
```

The previous options will create the kernel, but not the modules—those features of the kernel to be compiled into separate modules. To compile your modules, use the **make** command with the modules argument.

```
make modules
```

To install your modules, use the **make** command with the **modules_install** option. This installs the modules in the **/lib/modules/***version-num* directory, where *version-num* is the version number of the kernel. Making a backup copy of the old modules before you install the new ones may be advisable.

```
make modules_install
```

The commands for a simple compilation and installation are shown here:

```
make dep
make clean
make bzImage
make modules
make modules_install
make install
```

If you want, you could enter these all on one line, separating the commands with semicolons, as shown here:

```
make dep; make clean; make bzImage; make modules
make modules_install; make install
```

The following commands show a basic compilation and a manual installation. First, all previous binary files are removed with the **clean** option. Then the kernel is created using the **bzImage** option. This creates a kernel program called **bzImage** located in the **arch/i386/boot** directory. Then copy this kernel file to the **/boot** directory and give it the name **vmlinuz-*version***, where *version* is the kernel version. Then create a symbolic link called **/boot/vmlinuz** to the kernel **vmlinuz-*version*** file. Then create the modules and install the modules:

```
make dep
make clean
make bzImage
make modules
make modules_install
cp arch/i386/boot/bzImage /boot/vmlinuz-2.4.7-10
ln -s /boot/vmlinuz /boot/vmlinux-2.4.7-10
```

Instead of installing the kernel on your system, you can simply place it on a boot disk and boot your system from that disk. In that case, you just have to create a boot disk using the **bzdisk** option. Be sure a formatted floppy disk is in the floppy drive. You will still have to create and install your modules. Be sure that the **bzImage** file is small enough to fit on a floppy disk. If not, you will have to reconfigure your kernel, compiling as many features as possible as modules instead of as part of the kernel.

```
make clean
make bzImage
make bzdisk
make modules
make modules_install
```

Tip If you are experimenting with your kernel configurations, it may be safer to put a new kernel version on a boot disk, rather than installing on your system. If something goes wrong, you can always boot up normally with your original kernel still on your system.

LILO Configurations

If you are using a boot loader like GRUB or LILO, you can configure your system to enable you to start any of your installed kernels. As seen in the "Precautionary Steps" section, you can create an added entry in the boot loader configuration file for your old kernel. As you install new kernel versions, you could simply add more entries, enabling you to use any of the previous kernels. For example, you could install a developmental version of the kernel, along with a current stable version, while keeping your old version. In the image line for each entry, you would specify the filename of the kernel. Whenever you install the kernel on Red Hat using the RPM kernel package, the **/boot/vmlinuz** link is automatically changed to the new kernel. You can still create another boot loader entry for your older kernel. In the next example, the **grub.conf** file contains entries for two Linux kernels, one for the kernel installed with 7.2, 2.4.7-10, and one for a more recent kernel, 2.4.12. With GRUB, you only have to add a new entry for the new kernel. If you are using LILO, just add a new image segment for the new kernel in the **/etc/lilo.conf** file. Be sure to execute the **lilo** command to update LILO.

Table 34-1: Kernel Compile Options Used as Arguments to the make Command in /usr/src/linux

Configuration Tools	**Description**
config	Line-based interface for kernel configuration.
menuconfig	Screen-based interface for kernel configuration.
xconfig	X Window System interface for kernel configuration.
Maintenance Options	
checkhelp	Checks configuration for options not documented.
checkconfig	Checks source tree for missing header files.
clean	Removes old object files and dependencies.
mrproper	Performs a more complete removal of object files, including the kernel configuration file, **.config**. Usually run before a new patch.
Compiling Options	
zImage	Creates the kernel file called **zImage** located in the **/usr/src/linux/arch** or **arch/i386/boot** directory.
install	Creates the kernel and installs it on your system.
zlilo	Creates the kernel, installs it on your system, and runs LILO.
zdisk	Creates a kernel file and installs it on a floppy disk (creates a boot disk).
bzImage	Creates the kernel file and calls it **bzImage**.
bzlilo	Creates and installs the kernel and runs LILO.
bzdisk	Creates the kernel and installs it on a floppy disk (creates a boot disk).
Module Options	
modules	Creates kernel modules.
modules-install	Installs kernel modules in the **/lib/modules** directory.

/etc/grub.conf

```
# grub.conf generated by anaconda
#
#boot=/dev/hda
default=0
timeout=30
splashimage=(hd0,2)/boot/grub/splash.xpm.gz
title Red Hat Linux (2.4.12)
        root (hd0,2)
        kernel /boot/vmlinuz-2.4.12 ro root=/dev/hda3 hdc=ide-scsi
```

```
       initrd /boot/initrd-2.4.12.img
title Red Hat Linux (2.4.7-10)
       root (hd0,2)
       kernel /boot/vmlinuz-2.4.7-10 ro root=/dev/hda3 hdc=ide-scsi
       initrd /boot/initrd-2.4.7-10.img
title Windows XP
       root (hd0,0)
       imakeactive
       chainloader +1
```

Module RAM Disks

If your system requires certain modules to be loaded when you boot, you may have to create a RAM disk for them. For example, if you have a SCSI hard drive or CD-ROMs, the SCSI drivers for them are often held in modules that are loaded whenever you start up your system. These modules are stored in a RAM disk from which the startup process reads. If you create a new kernel that needs to load modules to start up, you must create a new RAM disk for those modules. When you create a new kernel, you also need to create its modules. You place the modules needed for startup, such as SCSI hard drive modules, in a new RAM disk. In the **lilo.conf** file, add an entry to load this RAM disk. You only need to create a new RAM disk if your kernel has to load modules at startup. If, for example, you use a SCSI hard drive, but you incorporated SCSI hard drive and CD-ROM support (including support for the specific model) directly into your kernel, you needn't set up a RAM disk. Support for IDE hard drives and CD-ROMs is already incorporated directly into the kernel.

If you need to create a RAM disk, you can use the **mkinitrd** command to create a RAM disk image file or create a RAM disk device. See the Man pages for **mkinitrd** and RAM disk documentation for more details. **mkinitrd** takes as its arguments the name of the RAM disk image file and the kernel that the modules are taken from. In the following example, a RAM disk image called **initrd-2.4.7-10.im** is being created in the **/boot** directory using modules from the 2.4.7-10 kernel. The 2.4.7-10 kernel needs to already be installed on your system and its modules created.

```
# mkinitrd /boot/initrd-2.4.7-10.img 2.4.7-10
```

In the **lilo.conf** segment for the new kernel, you would place an **initrd** entry specifying the new RAM disk.

```
image=/boot/vmlinuz-2.4.7-10
 label=linux
 root=/dev/hda3
 initrd=/boot/initrd-2.4.7-10.img
 read-only
```

Chapter 35: The X Window System and XFree86

Overview

Linux and Unix systems use the same standard underlying graphics utility known as the X Window System, also known as X or X11. This means that, in most cases, an X-based program can run on any of the window managers and desktops. X-based software is often found at Linux or Unix FTP sites in directories labeled **X11**. You can download these packages and run them on any window manager running on your Linux system. Some may already be in the form of Linux binaries that you can download, install, and run directly. Netscape is an example. Others are in the form of source code that can easily be configured, compiled, and installed on your system with a few simple commands. Some applications, such as Motif applications, may require special libraries.

The X Window System is designed for flexibility—you can configure it in various ways. You can run the X Window System on almost all the video cards currently available. The X Window System is not tied to any specific desktop interface. It provides an underlying set of graphical operations that user interface applications such as window managers, file managers, and even desktops can use. A window manager uses these operations to construct widgets for manipulating windows, such as scroll bars, resize boxes, and close boxes. Different window managers can construct them to appear differently, providing interfaces with different appearances. All window managers work on the X Window System. You can choose from a variety of different window managers, and each user on your system can run a different window manager, each using the same underlying X Window System graphic operations. You can even run X programs without any window or file managers.

To run the X Window System, you need to install an X Window server. A free version of X Window server software, known as XFree86*,* is used on most Linux systems, though commercial versions are available from MetroLink (**www.metrolink.com**) and Xi graphics. Once you install the XFree86 server, you must provide configuration information about your monitor, mouse, and keyboard. This information is then used in a configuration file called **/etc/X11/XF86Config**, which includes technical information best generated by an X Window System configuration program, such as Xconfigurator, xlizard, or XF86Setup. When you configured the X Window System when you installed your system, this file was automatically generated.

You can also configure your own X interface using the **.xinitrc** and **/etc/X11/ xinit/xinitrc** configuration files, where window managers, file managers, and initial X applications can be selected and started. And, you can use a set of specialized X commands to configure your root window, load fonts, or configure X Window System resources, such as setting the color of window borders. You can also download X utilities from online sources that serve as Linux mirror sites, usually in their **/pub/ Linux**/X11 directory. If you have to compile an X application, you may have to use special procedures, as well as install support packages. An official source for X Window System news, tools, and window managers is **www.X11.org**. Here you can find detailed information about X Window System features, along with compliant desktops and window managers.

The X Window System was developed and is maintained by The Open Group (TOG), a consortium of over a hundred companies, including Sun, HP, IBM, Motorola, and Intel (**www.opengroup.org**). Development is currently managed by the X.org group (**www.X.org**) on behalf of the TOG. X.org is a nonprofit organization that maintains the existing X Window System code. X.org periodically provides free official Window System update releases to the general public. It controls the development of the X11R6 specifications, working with appropriate groups to revise and release updates to the standard, as required. The newest release is currently X11R6.4. XFree86 is a free distributed version of X Window System servers used on most Linux systems. XFree86 incorporates X11R6.4 into its XFree86-4.0 release. You can find out more about XFree86 at **www.xfree86.org**.

The X Protocol

The X protocol was developed for Unix systems in the mid-1980s to provide a network-transparent graphical user interface. The X protocol organizes display operations into a client and server relationship, in which a client submits display requests to a server. The client is known as an X client, and the server as an X server. The client, in this case, is an application, and the server is a display. This relationship separates an application from the server. The application acts as a client sending requests to the server, which then does the actual work of performing the requested display operation. This has the advantage of letting the server interact with the operating system and its devices, whereas the application need know nothing of these details. An application operating as an X client can display on any system that uses an X server. In fact, a remote X client can send requests to have an X server on a local machine perform certain display operations. In effect, the X server/client relationship is inverted from the way we normally think of servers. Usually, several client systems access a single server. In the X server model, you would have each system operating as an X server that can access a single system that holds X client programs.

XFree86

The XFree86 Project (**www.xfree86.org**) is a nonprofit organization that provides free X Window System servers and supporting materials for several operating systems on PCs and other microcomputers. The X server, client programs, and documentation supplied by the XFree86 Project are commonly referred to as XFree86. The XFree86 server is available free and includes source code. The project is funded entirely by donations.

In releases 3.3 and earlier, XFree86 organized video card and monitor drivers into separate servers. You had to find the correct one to use and load it. With release 4.0, XFree86 now uses only one server, called the XFree86 X server, which includes all video card and monitor drivers. You only need to install the XFree86 X server package along with basic support packages such as those for fonts. The XFree86 X server will have support for given video cards and monitors implemented as static libraries or as modules it can load as needed. The XFree86 X server uses a built-in runtime loader, donated by Metro Link, that is operating system independent, though it is still hardware dependent. Currently, the XFree86 X server supports the Intel, Alpha, PowerPC, and Sparc platforms.

The XFree86 server supports a wide range of video cards and monitors, including monochrome, VGA, and Super VGA, and accelerated video cards. You can find a detailed listing of supported cards by checking the driver status for a particular release. To obtain information about a particular XFree86 release, just attach its release number to the **www.xfree86.org** site as in **www.xfree86.org/4.0** for information about release 4.0. To find out the driver status for release 4.1.0, go to **www.xfree86.org/4.1.0** and click the Driver Status link. This will list links for all video cards to pages detailing support for particular video card models. Also, you can consult the Man pages for the different driver types, such as nv for Nvidia cards, mga for Matrox, and ati for ATI graphics cards.

XFree86 configuration tools, such as Red Hat's Xconfigurator and XFree86's xf86cfg, make configuring your video card and monitor a simple process. They keep on hand an extensive list of video cards and monitors provided by Xfree86, from which you can select your own. If your video card or monitor is quite new, however, it may not be on this list. If this is the case, first check to see if a new release of XFree86 has come out. The new release may have support for your card or monitor. For example, release 4.1 now has support for NVidia GeForce3 and ATI Radeon cards. If your card is not supported, you will need to enter certain hardware specifications, such as horizontal and vertical sync frequencies for monitors. If you must do this, be careful to enter the correct information. The wrong settings could damage both your card and your monitor.

Be sure to check for new releases of XFree86 servers periodically at the XFree86 Web site. You can download the new releases from there or from your distribution's update sites, such as **ftp.redhat.com/updates** for Red Hat. It's always preferable to download from your Linux distribution sites, since those packages may be modified to work better with your system. The entire XFree86 software release includes the XFree86 X server and its modules along with several supporting packages such as those for fonts and configuration files. Table 35-1 lists the current XFree86 packages. If you are downloading from the Xfree86 site, you first download and run Xinstall.sh to determine which packages are appropriate for your system. XFree86 provides versions for different platforms on its FTP site at **ftp.xfree86.org**. For downloads from the XFree86 site, it is strongly recommended that you use the Xinstall.sh installer. Xinstall.sh will query for installation information and then download and install all needed XFree86 packages. For Red Hat systems, download and install the RPM packages from the Red Hat FTP site, instead of the ones at the XFree86 site.

Table 35-1: XFree86 Packages

Packages	**Description**
Xinstall.sh	The installer script
Extract	The utility for extracting tarballs
Xbin.tgz	X clients/utilities and runtime libraries
Xlib.tgz	Some data files required at runtime
Xman.tgz	Manual pages
Xdoc.tgz	XFree86 documentation
Xfnts.tgz	Base set of fonts
Xfenc.tgz	Base set of font encoding data
Xetc.tgz	Runtime configuration files
Xvar.tgz	Runtime data
Xxserv.tgz	XFree86 X server
Xmod.tgz	XFree86 X server modules
Optional Packages	Description
Xfsrv.tgz	Font server
Xnest.tgz	Nested X server
Xprog.tgz	X header files, config files, and compile-time libs
Xprt.tgz	X Print server
Xvfb.tgz	Virtual framebuffer X server

Table 35-1: XFree86 Packages

Packages	**Description**
Xf100.tgz	100-dpi fonts
Xfcyr.tgz	Cyrillic fonts
Xflat2.tgz	Latin-2 fonts
Xfnon.tgz	Some large bitmap fonts
Xfscl.tgz	Scalable fonts (Speedo and Type1)
Xhtml.tgz	HTML version of the documentation
Xps.tgz	PostScript version of the documentation
Xjdoc.tgz	Documentation in Japanese

In addition to the server, XFree86 includes support programs and development libraries. The entire XFree86 collection is installed in various directories, beginning with the pathname **/usr/X11R6**. Directories are here for X programs, development files, libraries, Man pages, and documentation. Configuration files are placed in the **/etc/X11** directory. Applications written to support *X* usually install in the **/usr/X11R6/bin** directory. You can also find the XFree86 servers and support programs here. Table 35-2 lists XFree86 configuration directories.

Table 35-2: XFree86 Directories

Directory	**Description**
/usr/X11R6/bin	Programs (X Window System clients and servers)
/usr/X11R6/include	Development files
/usr/X11R6/lib	Libraries
/usr/X11R6/man	Man pages
/usr/X11R6/lib/X11/doc	Documentation
/etc/X11	Configuration files
/usr/X11R6/lib/X11/	Contains subdirectories for window manager program functions

Note XFree86 now includes Direct Rendering Interface (DRI) and OPenGL support (GLX) for 3D cards like ATI, Matrox, and 3dfx (**dri.sourceforge.net**).

You can use X servers to run X Window System applications on a remote system. When you access a remote system, you can have the X server on that system generate a new display for you to run the remote X application. Every X server has a display name consisting of a hostname, a display number, and a screen number. These are used by an application to determine how to connect to the server and the screen it should use.

```
hostname:displaynumber.screennumber
```

The hostname is the host where the X server is physically located. The display number is the number of the display being managed by the X server. On a local workstation, there is usually only one display. However, on a multiuser system where several terminals (each with its own keyboard and mouse) are connected to a single system, each terminal is its own display with its own display number. This way, several users can be running X applications at the same time off the same X server. If your system has two or more monitors sharing the same keyboard and mouse, a different screen number would be applied to each monitor, though they would have the same display number.

The display a user is currently using is listed as the DISPLAY environment variable. On a single-user system, you will find that the display entry begins with a colon and is followed by a 0, as shown here. This indicates that the X server is on the local system (not a remote host) and has the display number of 0.

```
$ echo $DISPLAY
:0
```

To use a remote X application, you have to change the display name for the DISPLAY variable. You can do this manually by assigning a new hostname and display number to the variable, or you can use the **xon** script:

```
$ DISPLAY=rabbit.mytrek.com:0
$ export DISPLAY
```

You can also use the **-display** option when invoking an X application to specify the remote X server to use:

```
$ xterm -display rabbit.mytrek.com:0
```

XFree86 Configuration: /etc/X11/XF86Config

The XFree86 servers provide a wide range of hardware support, but it can be challenging to configure. You can consult the XFree86-HOWTO document at **www.linux.org** or in the **/usr/share/doc/HOWTO** directory for most distributions. There are also Man pages for XFree86 and XF86Config, and documentation and FAQs are available at **www.xfree86.org**. The configuration file used for your XFree86 server is called XF86Config, located in the **/etc/X11** directory. XF86Config contains all the specifications for your graphics card, monitor, keyboard, and mouse. To configure the XF86Config file, you need specific information on hand about your hardware. For your monitor, you must know the horizontal and vertical sync frequency ranges and bandwidth. For your graphics card, you have to know the chipset, and you may even need to know the clocks. For your mouse, you should know whether it is Microsoft-compatible or some other brand, such as Logitech. Also, know the port to which your mouse is connected.

Note XFree86 4.x uses the **/etc/XF86Config-4** file, instead of **/etc/XF86Config**, if it exists. Red Hat generates both files, but the **/etc/XF86Config-4** file takes precedence and uses XFree86 4.x commands.

Although you could create and edit the file directly, using a configuration utility such as Xconfigurator or xf86config is better. (Table 35-3 lists these various configuration tools.) With these, you simply answer questions about your hardware or select options on the dialog window, and the program generates the appropriate **/etc/X11/XF86Config** file. xf86config provides line-mode prompts where you type responses or enter a menu selection, and it provides explanations of each step. You can run it from any shell command line. Xconfigurator uses a cursor-based screen that also operates on a shell command line. You can use arrow keys, TAB, and the ENTER key to make your selections. xf86config also attempts to detect your card automatically, or you can select your monitor from a predetermined list.

Table 35-3: X Window System Configuration Tools

Tool	Description
xf86cfg	XFree86 screen-based X Window System configuration tool
XFree86 -configure	XFree86 X Window System configuration tool that is built into the XFree86 X server
XF86Setup	GUI X Window System configuration tool; use after installation process
Xconfigurator	Screen-based X Window System configuration tool (used in Red Hat install procedure)
xf86config	XFree86 command line X Window System configuration tool; requires no screen-based support
/etc/X11/XF86Config	The X Window System configuration file; edited by the configuration tools

If you have problems configuring with a configuration utility, however, you can use the **XFree86** command with the **–configure** option to generate a version for you. With the **–configure** option, XFree86 will probe your video card, keyboard, and mouse, and then generate an XF86Config file automatically, naming it **XF86Config.new**. It will also display a report on the results of its probe. This approach can be helpful if you have difficulty installing a new video card:

```
XFree86 –configure
```

You can check out the configuration first by trying to run it with the X Window System. Use the **–xf86config** option to use the **XF86Config.new** file instead of the **/etc/XF86Config** file:

```
XFree86 –xf86config /root/XF86Config.new
```

Once this version of the configuration file is working, you can replace your current **/etc/XF86Config** file with it.

The **/etc/X11/XF86Config** file is organized into several parts, as shown here. You can find a detailed discussion of all these sections and their entries in the XF86Config Man page. All of these are set by the XF86Setup program. For example, the Monitor screen generates the Monitor section in the XF86Config file, the Mouse screen generates the Input Device section for the mouse, and so on. A section in the file begins with the keyword **Section,** followed by the name of the section in quotes. The section ends with the term **EndSection**. Comments have a # sign at the beginning of the line. The different kinds of sections are listed here.

Section	Description
Files	Directories for font and rgb files
Module	Dynamic module loading
ServerFlags	Miscellaneous options

Section	Description
Input Device	Mouse and keyboard configuration
Monitor	Monitor configuration (set horizontal and vertical frequencies)
Device	Video card configuration
Screen	Configure display, setting virtual screen, display colors, screen size, and other features
ServerLayout	Specify layout of screens and input devices

Entries for each section begin with a data specification, followed by a list of values.

With release 4.0, many former data specifications are now implemented using the Option entry. You enter the keyword **Options** followed by the data specification and its value. For example, the keyboard layout specification, XkbLayout, is now implemented using an Options entry as shown here:

```
Option "XkbLayout" "us"
```

Although you can directly edit the file using a standard text editor, relying on the setup programs such as XF86Setup to make changes is always best. You won't ever have to touch most of the sections, but in some cases, you want to make changes to the Screen section, located at the end of the file. To do so, you would edit the file and add or change entries in the Screen section. In the Screen section, you can configure your virtual screen display and set the number of colors supported. Because the Screen section is the one you would most likely change, it is discussed first, even though it comes last, at the end of the file.

Screen

A Screen section begins with an Identifier entry to give a name to this Screen. After the Identifier entry, the Device and Monitor entries specify the monitor and video card you are using. The name given in the Identifier entry in these sections is used to reference those components.

```
Section "Screen"
 Identifier "Screen0"
 Device "Matrox|MGA G400 AGP"
 Monitor "C1025"
 DefaultDepth 24
 Subsection "Display"
 Depth 32
 Modes "1024x768"
 EndSubSection
 Subsection "Display"
 Depth 24
 Modes "1024x768"
 EndSubSection
EndSection
```

The Screen section has Display subsections, one for each depth supported. Whereas the previous sections configured hardware, the Display subsection configures display features, such as the number of colors displayed and the virtual screen size. Two main entries exist: Depth and Modes. The Depth entry is the screen resolution: 8, 16, and 24. You can add the

DefaultDepth entry to set the default color depth to whatever your X server supports: 8 for 256K, 16 for 32K, and 24 for 16M. Modes are the modes allowed given the resolution. You can also add to the Virtual entry to specify the size of the virtual screen. You can have a virtual screen larger than your display area. When you move your mouse to the edge of the displayed screen, it scrolls to that hidden part of the screen. This way, you can have a working screen much larger than the physical size of your monitor. The physical screen size for a 17-inch monitor is usually 1,024 x 768. You could set it to 1,152 x 864, a 21-inch monitor size, with a Virtual entry.

Any of these features in this section can be safely changed. In fact, to change the virtual screen size, you must modify this section. Other sections in the XF86Config file should be left alone, however, unless you are certain of what you are doing.

Files, Modules, and ServerFlags

The Files section lists different directories for resources that XFree86 needs. For example, to specify the location where **rgb** color data is listed, a line begins with the data specification **RgbPath,** followed by the pathname for that **rgb** color data file. For fonts, you can specify either font files or a font server. In earlier versions of the X Window System, this section mostly listed the fonts available on your system. A font entry would begin with the data specification **FontPath** and was followed by the pathname for that font. A sample of these entries is shown here:

```
RgbPath "/usr/X11R6/lib/X11/rgb"
FontPath "/usr/X11R6/lib/X11/fonts/misc:unscaled"
```

To specify a font server, you enter the transport type, the hostname, and the port it is listening on (this is usually 7100). If the server is on your own system, you can leave the hostname blank.

```
transport/hostname:port
```

The *transport* can be either tcp for a TCP/IP connection (Internet) or unix for a Unix domain socket. On most systems, the **xfs** server manages fonts. Font paths for the **xfs** server are listed in the **/etc/X11/fs/config** file. A sample entry for the font server is shown here for a local font server listening on port 7100.

```
FontPath "unix/:7100"
```

If no FontPaths are specified, the X server falls back on default font paths already compiled into the X server (see the XF86Config Man page for more details).

The Module section specifies modules to be dynamically loaded, and the Load entry loads a module. It is used to load server extension modules and font modules. This is a feature introduced with version 4.0 that allows X server components that extend the functionality of the X server to be loaded as modules. This feature provides for easy updating, letting you upgrade modules without having to replace the entire X server. For example, the extmod module contains miscellaneous extensions to enable commonly used functions in the X server. In the following example, the extmod module that contains a set of needed extensions is loaded. See the XF86Config Man page for more details.

```
Load "extmod"
```

Several flags can be set for the XFree86 server. With version 4.0, these are now implemented as Options. (You can find a complete listing in the XF86Config Man page.) For example, the **BlankTime** value specifies the inactivity timeout for the screen saver. **DontZap** disables the use of CTRL-ALT-BACKSPACE to shut down the server. **DontZoom** disables switching between graphic modes. You create an Option entry with the flag as the option. The following example sets the server flag for the screen saver inactivity timeout:

```
Option "BlankTime " "30"
```

Intput Device

With 4.0, the Input Device section replaces the previous Keyboard, Pointer, and XInput sections. To provide support for an input device such as a keyboard, you create an Input Device section for it and enter Identifier and Driver entries for the device. For example, the following entry creates an Input Device section for the keyboard:

```
Section "Input Device"
 Identifier "keyboard 1"
 Driver "keyboard"
```

Any features are added as Options, such as keyboard layout or model. A large number of options exist for this section. Consult the XF86Config Man pages for a complete listing. The following example shows an entire keyboard entry with autorepeat, keyboard model (XkbModel), and keyboard layout (XkbLayout) options entered:

```
Section "InputDevice"
 Identifier "Keyboard 1"
 Driver "keyboard"
 Option "AutoRepeat" "500 5"
 Option "XkbModel" "pc104"
 Option "XkbLayout" "us"
EndSection
```

You create an Input Device section for your mouse and any other pointer devices. This section has only a few entries, with some tailored for specific types of mice. Features are defined using Option entries. The Protocol option specifies the protocol your mouse uses, such as PS/2, Microsoft, or Logitech. The Device option is the pathname for the mouse device. The following example shows a standard Pointer section for a three-button PS/2 mouse. The device file is **/dev/mouse**.

```
Section "InputDevice"
 Identifier "Mouse 1"
 Driver "mouse"
 Option "Protocol" "PS/2"
 Option "Device" "/dev/mouse"
 Option "Emulate3Buttons" "off"
EndSection
```

The following is a listing of the Pointer section options:

Option	**Description**

Option	Description
Protocol	Mouse protocol (Check the XF86Config Man page for complete listing)
Device	Device path, such as /dev/mouse or /dev/cua0
BaudRate	Baud rate for serial mouse
Emulate3Buttons	Enables two-button mouse to emulate a third button when both left and right buttons are pressed at once
ChordMiddle	Three-button mouse configuration on some Logitech mice
ClearDTR and ClearRTS	Clear DTR and RTS lines, valid only for Mouse Systems mice
SampleRate	Set the sampling rate (Logitech)

Monitor

A Monitor section should exist for each monitor used on your system. The vertical and horizontal frequencies must be accurate, or you can damage your monitor. A Monitor section begins with entries that identify the monitor, such as vendor and model names. The HorizSync and VerRefresh entries are where the vertical and horizontal frequencies are specified. Most monitors can support a variety of resolutions. Those resolutions are specified in the Monitor section by ModeLine entries. A ModeLine entry exists for each resolution. The ModeLine entry has five values, the name of the resolution, its dot clock value, and then two sets of four values, one for the horizontal timing and one for the vertical timing, ending with flags. The flags specify different characteristics of the mode, such as Interlace, to indicate the mode is interlaced, and +hsync and +vsync to select the polarity of the signal.

```
ModeLine "name" dotclock horizontal-freq vertical-freq flags
```

A sample of a ModeLine entry is shown here. Leaving the entire Monitor section alone is best; rely, instead, on the entries generated by XF86Setup.

```
Modeline "800x600" 50.00 800 856 976 1040 600 637 643 666 +hsync +vsync
```

Commonly used entries for the Monitor section are listed here:

Option	Description
Identifier	A name to identify the monitor
VendorName	Manufacturer
ModelName	The make and model
HorizSync	The horizontal frequency; can be a range or series of values
VerRefresh	Vertical refresh frequency; can be a range or series of values
Gamma	Gamma correction
ModeLine	Specifies a resolution with dotclock, horizontal timing, and vertical timing for that resolution

A sample Monitor section is shown here:

```
Section "Monitor"
 Identifier "C1025"
 VendorName "Unknown"
 ModelName "Unknown"
 HorizSync 30 - 95
 VertRefresh 50 - 160
EndSection
```

Device

The Device section specifies your video card. It begins with an Identifier entry and an entry for the video card driver. The following example creates an Identifier for a Matrox card called "MGA 1" and then specifies that the **mga** driver is to be used for it:

```
Identifier " Matrox|MGA G400 AGP"
Driver "mga"
```

Further entries identify the card, such as VendorName, BoardName, and Chipset. The amount of video RAM is indicated in the VideoRam entry. The Clocks entry lists your clock values. Many different entries can be made in this section, such as Ramdac for a Ramdac chip, if the board has one, and MemBase for the base address of a frame buffer, if it is accessible. See the XF86Config Man pages for a detailed list and descriptions.

Although you could safely change a VideoRam entry—for example, if you added more memory to your card—changing the Clocks entry is not safe. If you get the clock values wrong, you could easily destroy your monitor. Rely on the clock values generated by XF86Setup or other XFree86 setup programs. If the clock values are missing, it means that the server will automatically determine them. This may be the case for newer cards. A sample Device entry is shown here:

```
Section "Device"
 Identifier "Matrox|MGA G400 AGP"
 Driver "mga"
 BoardName "Unknown"
EndSection
```

ServerLayout

A ServerLayout section has been added with release 4.0 to let you specify the layout of the screens and the selection of input devices. The ServerLayout sections may also include options that are normally found in the ServerFlags section. You can set up several ServerLayout sections and select them from the command line. The following example shows a simple ServerLayout section for a basic configuration:

```
Section "ServerLayout"
 Identifier "XFree86 Configured"
 Screen 0 "Screen0" 0 0
 InputDevice "Mouse0" "CorePointer"
 InputDevice "Keyboard0" "CoreKeyboard"
EndSection
```

X Window System Command Line Arguments

You can start up any X Window application either within an **.xinitrc** or **.xsession** script or on the command line in an Xterm window. Some distributions, including Red Hat, place X Window startup applications in an **.Xclients** file that is read by the **.xinitrc** script. Most X Window applications take a set of standard X Window arguments used to configure the window and display the application uses. You can set the color of the window bars, give the window a specific title, and specify the color and font for text, as well as position the window at a specific location on the screen. Table 35-4 lists these X Window arguments. They are discussed in detail in the XMan pages, **man X**.

Table 35-4: Configuration Options for X Window System–Based Applications

X Window Application Configuration Argument	See the XMan pages for detailed explanations
-bw *num*	Border width of pixels in frame
-bd *color*	Border color
-fg *color*	Foreground color (for text or graphics)
-bg *color*	Background color
-display *display-name*	Displays client to run on; displays name consisting of hostname, display number, and screen number (see XMan pages)
-fn font	Font to use for text display
-geometry *offsets*	Location on screen where X Window application window is placed; offsets are measured relative to screen display
-iconic	Starts application with icon, not with open window
-rv	Switches background and foreground colors
-title string	Title for the window's title bar
-name *string*	Name for the application
-xrm *resource-string*	Specifies resource value
-xnllanguage *lang*	Set the language used to resolve filenames

One commonly used argument is **-geometry**. This takes an additional argument that specifies the location on the screen where you want an application's window displayed. In the next example, the xclock X Window application is called with a geometry argument. A set of up to four numbers specifies the position. The value +0+0 references the upper left-hand corner. There, you see the clock displayed when you start up the X Window System. The value –0–0 references the upper right-hand corner.

```
& xclock -geometry +0+0 &
```

With the **-title** option, you can set the title displayed on the application window. Notice the use of quotes for titles with more than one word. You set the font with the **-fn** argument and the text and graphics color with the **-fg** argument. **-bg** sets the background color. The following example starts up an Xterm window with the title "My New Window" in the title bar. The text and graphics color is green, and the background color is gray. The font is Helvetica.

```
$ xtessrm -title "My New Window" -fg green -bg gray -fn
/usr/fonts/helvetica &
```

X Window System Commands and Configuration Files

The X Window System uses several configuration files as well as X commands to configure your X Window System. Some of the configuration files belong to the system and should not be modified. Each user can have his or her own set of configuration files, however, such as **.xinitrc**, **.xsession**, and **.Xresources**, that can be used to configure a personalized X Window interface. Red Hat also uses an **.Xclients** file to hold X Window startup applications. These configuration files are automatically read and executed when the X Window System is started up with either the **startx** command or an X display manager, such as xdm or gdm. Within these configuration files, you can execute X commands used to configure your system. With commands such as **xset** and **xsetroot**, you can add fonts or control the display of your root window. Table 35-5 provides a list of X Window System configuration files and commands. You can obtain a complete description of your current X configuration using the **xdypinfo** command. The XMan pages provide a detailed introduction to the X commands and configuration files.

X Resources

Several X commands, such as **xrdb** and **xmodmap,** configure your X Window interface. X Window graphic configurations are listed in a resource file called **.Xresources**. Each user can have a customized **.Xresources** file in his or her home directory, configuring the X Window System to particular specifications. The **.Xresources** file contains entries for configuring specific programs, such as the color of certain widgets. A systemwide version called **/etc/X11/xinit/.Xresources** also exists. (Notice that, unlike **/etc/X11/xinit/xinitrc**, a period is before **Xresources** in the **/etc/X11/xinit/.Xresources** filename.) The **.Xdefaults** file is a default configuration loaded by all programs, which contains the same kind of entries for configuring resources as **.Xresources**. An **.Xdefaults** file is accessible by programs on your system, but not by those running on other systems. The **/usr/X11R6/ lib/X11/app-defaults** directory holds files that contain default resource configurations for particular X applications, such as Xterm, Xclock, and Xmixer. The Xterm file holds resource entries specifying how an Xterm window is displayed. You can override any of these defaults with alternative entries in an **.Xresources** file in your home directory. You can create an **.Xresources** file of your own in your home directory and add resource entries to it. You can also copy the **/etc/X11/xinit/.Xresources** file and edit the entries there or add new ones of your own.

Configuration is carried out by the **xrdb** command, which reads both the system's **.Xresources** file and any .**Xresources** or .**Xdefaults** file in your home directory. The **xrdb** command is currently executed in the **/etc/X11/xinit/xinitrc** script and the **/etc/X11/xdm/Xsession** script. If you create your own **.xinitrc** script in your home directory, be sure it executes the **xrdb** command with at least your own **.Xresources** file or the **/etc/X11/xinit/.Xresources** file (preferably both). You can ensure this by simply using a copy of the system's **xinitrc** script as your own **.xinitrc** file, and then modifying that copy as you want. See the Man pages on **xrdb** for more details on resources. Also, you can find a more detailed discussion of Xresources, as well as other X commands, in the Man pages for *X.*

An entry in the **.Xresources** file consists of a value assigned to a resource, a class, or a group of resources for an application. Usually, resources are used for widgets or classes of widgets in an application. The resource designation typically consists of three elements separated by periods: the application, an object in the application, and the resource. The entire designation is terminated by a colon, and the value follows. For example, suppose you want to change the color of the hour hand to blue in the oclock application. The application is oclock, the object is clock, and the resource is hour: oclock.clock.hour. This entry looks like this:

```
oclock.clock.hour: blue
```

The object element is actually a list of objects denoting the hierarchy leading to a particular object. In the oclock example, only one object exists, but in many applications, the object hierarchy can be complex. This requires a lengthy set of objects listed to specify the one you want. To avoid this complexity, you can use the asterisk notation to reference the object you want directly, using an asterisk in place of the period. You only need to know the name of the resource you want to change. The following example sets the oclock minute and hour hands to green:

```
oclock*hour: green
oclock*minute: green
```

You can also use the asterisk to apply a value to whole classes of objects. Many individual resources are grouped into classes. You can reference all the resources in a class by their class name. Class names begin with an uppercase character. In the Xterm application, for example, the background and pointer color resources are both part of the Background class. The reference **XTerm*Background** would change all these resources in an Xterm window. However, any specific references always override the more general ones.

You can also use the asterisk to change the values of a resource in objects for all your applications. In this case, you place an asterisk before the resource. For example, to change the foreground color to red for all the objects in every application, you enter

```
*foreground: red
```

If you want to change the foreground color of the scroll bars in all your applications, you use

```
*scrollbar*foreground: blue
```

The **showrgb** command lists the different colors available on your system. You can use the descriptive name or a hexadecimal form. Values can also be fonts, bitmaps, and pixmaps. You could change the font displayed by certain objects in, or for, graphic applications as well as change background or border graphics. Resources vary with each application. Applications may support different kinds of objects and the resources for them. Check the Man pages and documentation for an application to learn what resources it supports and the values accepted for it. Some resources take Boolean values that can turn features on or off, while others can specify options. Some applications have a default set of resource values that is automatically placed in your system's .**Xresources** or .**Xdefaults** files.

The **Xmodmap** file holds configurations for your input devices, such as your mouse and keyboard (for example, you can bind keys such as BACKSPACE or reverse the click operations of your right and left mouse buttons). The **Xmodmap** file used by your display manager is in the display manager configuration directory, such as **/etc/X11/xdm**, whereas the one used by **startx** is located in **/etc/X11/xinit**. Each user can create a custom **.Xmodmap** file in his or her home directory to configure the system's input devices. This is helpful if users connect through their own terminals to your Linux system. The **.Xmodmap** file is read by the **xmodmap** command, which performs the configuration. The **xmodmap** command first looks for an **.Xmodmap** file in the user's home directory and uses that. If no **.Xmodmap** is in the home directory, it uses the one for your display manager or **startx** command. You see entries for the **xmodmap** command in the **/etc/X11/xinit/xinitrc** file and the display manager's **Xsession** file. If you have your own **.xinitrc** or **.xsession** script in your home directory, it should execute the **xmodmap** command with either your own **.Xmodmap** file or the system's **Xmodmap** file. See the Man pages on **xmodmap** for more details.

X Commands

Usually, an **.xinitrc** or **.xssesion** script has X Window System commands, such as **xset** and **xsetroot**, used to configure different features of your X Window session. The **xset** command sets different options, such as turning on the screen saver or setting the volume for the bell and speaker. You can also use **xset** to load fonts. See the **xset** Man pages for specific details. With the **b** option and the **on** or **off** argument, **xset** turns your speaker on or off. The following example turns on the speaker:

```
xset b on
```

You use **xset** with the **-s** option to set the screen saver. With the **on** and **off** arguments, you can turn the screen saver on or off. Two numbers entered as arguments specify the length and period in seconds. The length is the number of seconds the screen saver waits before activating and the period is how long it waits before regenerating the pattern.

The **xsetroot** command enables you to set the features of your root window (setting the color or displaying a bitmap pattern—you can even use a cursor of your own design). Table 35-5 lists the different **xsetroot** options. See the Man pages for **xsetroot** for options and details. The following **xsetroot** command uses the **-solid** option to set the background color of the root window to blue:

```
xsetroot -solid blue
```

Table 35-5: X Window System Commands

X Window Command	Explanation
xterm	Opens a new terminal window
xset	Sets X Window options; see Man pages for complete listing **-b** Configures bell **-c** Configures key click **+fp** *fontlist* Adds fonts **-fp** *fontlist* Removes fonts **led** Turns on or off keyboard LEDs **m** Configures mouse **p** Sets pixel color values **s** Sets the screen saver **q** Lists current settings
xsetroot	Configures the root window

Table 35-5: X Window System Commands

X Window Command	Explanation
	-cursor *cursorfile maskfile* Sets pointer to bitmap pictures when pointer is outside any window **-bitmap** *filename* Sets root window pattern to bitmap **-gray** Sets background to gray **-fg** *color* Sets color of foreground bitmap **-bg** *color* Sets color of background bitmap **-solid** *color* Sets background color **-name** *string* Sets name of root window to string
xmodmap	Configures input devices; reads the **.Xmodmap** file **-pk** Displays current keymap **-e** *expression* Sets key binding keycode NUMBER = KEYSYMNAME Sets key to specified key symbol keysym KEYSYMNAME = KEYSYMNAME Sets key to operate the same as specified key pointer = NUMBER Sets mouse button codes
xrdb	Configures X Window resources; reads the **.Xresources** file
xdm	X Window display manager; runs the XFree86 server for your system; usually called by **xinitrc**
Startx	Starts X Window by executing **xinit** and instructing it to read the **xinitrc** file.
xfs *config-file*	The X Window font server
mkfontdir *font-directory*	Indexes new fonts, making them accessible by the font server
xlsfonts	Lists fonts on your system
xfontsel	Displays installed fonts
xdpyinfo	Lists detailed information about your X Window configuration
xinit	Starts X Window, first reading the system's **xinitrc** file; when invoked from **startx**, it also reads the user's **.Xclients** file; **xinit** is not called directly, but through **startx**
xmkmf	Creates a **Makefile** for an X Window application using the application's **Imakefile;** invokes **imake** to generate the **Makefile** (never invoke **imake** directly)
xauth	Reads **.Xauthority** file to set access control to a user account through xdm from remote systems

Fonts

Your X Window System fonts are located in a directory called **/usr/X11R6/lib/X11/ fonts**. X Window System fonts are loaded using the **xfs** command. **xfs** reads the **/etc/X11/fs/config** configuration file that lists the font directories in an entry for the term **catalogue**. The XMan pages provide a detailed discussion on fonts. To install a set of fonts automatically, place them in a directory whose path you can add to the catalogue entry. You can also separately install a particular font with the **xset** command and its **+fp** option. Fonts for your system are specified in a font path. The *font path* is a set of filenames, each holding a font. The filenames include their complete path. An example of the catalogue entry in the **/etc/X11/config** file follows. This is a comma-delimited list of directories. These are directories where the X Window System first looks for fonts.

```
catalogue = /usr/X11R6/lib/X11/fonts/misc/,
/usr/X11R6/lib/X11/fonts/Speedo/,/usr/X11R6/lib/X11/fonts/Type1/,
/usr/X11R6/lib/X11/fonts/75dpi/,/usr/X11R6/lib/X11/fonts/100dpi/
```

Before you can access newly installed fonts, you must first index them with the **mkfontdir** command. From within the directory with the new fonts, enter the **mkfontdir** command. You can also use the directory path as an argument to **mkfontdir**. After indexing the fonts, you can then load them using the **xset** command with the **fp rehash** option. To have the fonts automatically loaded, add the directory with the full pathname to the catalogue entry in the **xfs** configuration file. The following shows how to install a new font and then load it:

```
$ cp newfont.pcf ~/myfonts
$ mkfontdir ~/myfonts
$ xset fp rehash
```

Within a font directory, several special files hold information about the fonts. The **fonts.dir** file lists all the fonts in that directory. In addition, you can set up a **fonts.alias** file to give other names to a font. Font names tend to be long and complex. A **fonts.scale** file holds the names of scalable fonts. See the Man pages for **xfs** and **mkfontdir** for more details.

With the **xset +fp** and **-fp** options, you can specifically add or remove particular fonts. The **fn** option with the **rehash** argument then loads the fonts. With the default argument, the default set of fonts is restored. The **+fp** option adds a font to this font path. For your own fonts, you can place them in any directory and specify their filenames, including their complete path. The next example adds the **myfont** font in the **/usr/local/fonts** directory to the font path. Then, the **fp** option with the **rehash** argument loads the font.

```
xset +fp /usr/local/fonts/myfont
xset fp rehash
```

To remove this font, use **xset -fp /usr/home/*myfont*** and follow it with the **xset fp rehash** command. If you want to reset your system to the set of default fonts, enter the following:

```
xset fp default
xset fp rehash
```

With **xlsfonts**, you can list the fonts currently installed on your system. To display an installed font to see what it looks like, use **xlslfonts**. You can browse through your fonts, selecting the ones you like.

Table 35-5 lists common X Window System commands, whereas Table 35-6 lists the configuration files and directories associated with the X Window System.

Table 35-6: X Window System Configuration Files

Configuration File	Explanation
.Xmodmap	User's X Window input devices configuration file.

Table 35-6: X Window System Configuration Files

Configuration File	**Explanation**
.Xresources	User's X Window resource configuration file.
.Xdefaults	User's X Window user-specific resource configuration file.
.xinitrc	User's X Window configuration file read automatically (by **xinit**, if it exists).
.Xclients or .Xsessions	User's X Window configuration file (used on Red Hat and other Linux distributions).
.Xauthority	User's access controls through xdm GUI login interface.
/usr/X11R6/	Directory where the X Window System release 6 commands, applications, and configuration files are held.
/usr/X11R6/lib/X11/	Link to the **/etc/X11** directory that holds X Window System release 6 configuration file and subdirectories.
/etc/X11/	Directory that holds X Window System release 6 configuration file and subdirectories.
/usr/X11R6/lib/X11/	Directory that holds X Window System configuration files and subdirectories for the version currently installed on your system. On Red Hat, this is a link to the **/etc/X11** directory.
/etc/X11/xinit/xinitrc	System X Window initialization file; automatically read by **xinit**
/etc/X11/xinit/Xclients	System X Window configuration file (used on Red Hat and other Linux distributions).
/etc/X11/xinit/.Xresources	System X Window resources file; read by **xinitrc**.
/etc/X11/xinit/.Xmodmap	System X Window input devices file; read by **xinitrc**.
/etc/X11/rgb.txt	X Window colors. Each entry has four fields: the first three fields are numbers for red, green, and blue; the last field is the name given to the color.

X Window System Startup Methods: startx and Display Managers

You can start up your X Window System in two different ways. You can start Linux with the command line interface and then, once you log in, use the **startx** command to start the X Window System and your window manager and desktop. You can also use a display manager that automatically starts the X Window System when you boot your computer, displaying a login window and a menu for selecting the window manager or desktop you want to use. Options for shutting down your system are also there. Currently, you can use three display managers: The K Display Manager is a display manager provided with the KDE. The Gnome Display Manager comes with the Gnome desktop. The X Display Manager is the original display manager used on Linux system.

Each method uses its own startup script. The **startx** command uses the **xinit** command to start the X Window System; its startup script is **/etc/X11/xinit/xinitrc**. Startup scripts for display managers are found in their respective directories. For xdm, the startup script is

/etc/X11/xdm/Xsession. gdm has its own configuration directory, **/etc/X11/gdm**. Here you find files for configuring their login window and menus. The gdm application as currently implemented on Red Hat uses the Xdm **Xsession** script.

As an enhancement to either **startx** or a display manager, you can use the X session manager (xsm). You can use it to launch your X Window System with different sessions. A *session* is a specified group of X applications. Starting with one session might start Gnome and Netscape, while starting with another might start KDE and KOffice. You can save your session while you are using it or when you shut down. The applications you are running become part of a saved session. When you start, xsm displays a session menu for you to choose from, listing previous sessions you saved. For xsm to work, it must be the last entry in your **.xsessions** or **.Xclients** file, and you shouldn't have any other applications started in these files.

startx, xinit, and .xinitrc

The X Window System can be started from the command line interface using the **xinit** command. You do not invoke the **xinit** command directly, but through the **startx** command, which you always use to start the X Window System. Both of these commands are found in the **/usr/X11R6/bin** directory, along with many other X-based programs. The **startx** command is a shell script that executes the **xinit** command. The **xinit** command, in turn, first looks for an X Window System initialization script called **.xinitrc**, in the user's home directory. If no **.xinitrc** script is in the home directory, **xinit** uses **/etc/X11/xinit/xinitrc** as its initialization script. Both **.xinitrc** and **/etc/X11/ xinit/xinitrc** have commands to configure your X Window server and to execute any initial X commands, such as starting up the window manager. You can think of the **/etc/X11/xinit/xinitrc** script as a default script. In addition, many systems use a separate file named **Xclients,** where particular X applications, desktops, or window managers can be specified. These entries can be directly listed in an **xinitrc** file, but a separate file makes for a more organized format. The **Xclients** files are executed as shell scripts by the **xinitrc** file. A user version, as well as a system version, exists: **.Xclients** and **/etc/X11/init/Xclients**. On Red Hat systems, the user's home directory is checked for the **.Xclients** file and, if missing, the **/etc/X11/xinit/Xclients** file is used.

Most distributions, including Red Hat, do not initially set up any **.xinitrc** or **.Xclients** scripts in any of the home directories. These must be created by a particular user who wants one. Each user can create a personalized **.xinitrc** script in her or his home directory, configuring and starting up the X Window System as wanted. Until a user sets up an **.xinitrc** script, the **/etc/X11/xinit/xinitrc** script is used, and you can examine this script to see how the X Window System starts. Certain configuration operations required for the X Window System must be in the **.xinitrc** file. For a user to create his or her own **.xinitrc** script, copying the **/etc/X11/xinit/xinitrc** first to the home directory and naming it **.xinitrc** is best. Then, each user can modify the particular **.xinitrc** file as required. (Notice the system **xinitrc** file has no preceding period in its name, whereas the home directory **.xinitrc** file set up by a user does have a preceding period.) The following example shows a simplified version of the system **.xinitrc** file that starts the Window Maker window manager and an Xterm window. System and user **.Xresources** and **.Xmodmap** files are executed first to configure the X Window System.

.xinitrc

```
#!/bin/sh
```

```
userresources=$HOME/.Xresources
usermodmap=$HOME/.Xmodmap
sysresources=/usr/X11R6/lib/X11/xinit/.Xresources
sysmodmap=/usr/X11R6/lib/X11/xinit/.Xmodmap

# merge in defaults and keymaps
if [ -f $sysresources ]; then
      xrdb -merge $sysresources
fi
if [ -f $sysmodmap ]; then
      xmodmap $sysmodmap
fi
if [ -f $userresources ]; then
      xrdb -merge $userresources
fi
if [ -f $usermodmap ]; then
      xmodmap $usermodmap
fi
# start some nice programs
      xterm &
      exec wmaker &
```

On Red Hat, if a user only wants to add startup applications, the user can simply create an **.Xclients** file instead of a complete **.xinitrc** file. Be sure commands are in the system **xinitrc** file to check for and run a user's **.Xclients** file. The following example shows the code used in the Red Hat **xinitrc** file to execute a user's **.Xclients** script and, failing that, the system **Xclients** script. If this fails, the FVWM2 window manager is started and, if that fails, the twm file manager is started.

```
if [ -f $HOME/.Xclients ]; then
           exec $HOME/.Xclients
elif [ -f /etc/X11/xinit/Xclients ]; then
           exec /etc/X11/xinit/Xclients
else
 # failsafe settings. Although we should never get here
 xclock -geometry 100x100-5+5 &
 xterm -geometry 80x50-50+150 &
   if [ -f /usr/X11R6/bin/fvwm2 ]; then
           exec fvwm2
           else
           exec twm
   fi
fi
```

Display Managers: xdm and gdm

When a system configured to run a display manager starts up, the X Window System starts up immediately and displays a login dialog box. The dialog box prompts the user to enter a login name and a password. Once they are entered, a selected X Window interface starts up—say, with Gnome, KDE, or some other desktop or window manager. When the user quits the window manager or desktop, the system returns to the login dialog box and remains there until another user logs in. You can shift to a command line interface with the CTRL-ALT-F1 keys and return to the display manager login dialog box with CTRL-ALT-F7. A display manager can do much more than provide a GUI login window. You can also use it to control access to different hosts and users on your network. The **.Xauthority** file in each user's home directory contains authentication information for that user. A display manager like xdm supports the X Display Manager Control Protocol (XDMCP). They were originally designed for systems like workstations that are continually operating, but they are also used to start up X Window automatically on single-user systems when the system boots.

A display manager is automatically run when your system starts up at runlevel 5. Recall that your system can run at different runlevels; for example, runlevel 3 is the standard multiuser level, whereas runlevel 2 is a non-network user level, and runlevel 1 is a system administration level. Runlevel 5 is the same as the standard multiuser level (runlevel 3), except it automatically starts up the X Window System on connected machines and activates the display manger's login screen.

During installation, you could choose whether you wanted to start with the display manager (runlevel 5) or not (runlevel 3). In this case, your system automatically starts at runlevel 5, activating the display manager. If, instead, you are starting with a standard line-mode login prompt (runlevel 3), you can manually change to the display manager by changing your runlevel. To do this, you can specify your runlevel with the telinit administration utility (see Chapter 4). The following command changes to runlevel 3, the command line:

```
telinit 3
```

This command will change to runlevel 5, the graphical login:

```
telinit 5
```

To make a runlevel the default, you have to edit the **/etc/inittab** file as described in Chapter 28.

Xsession

A display manager refers to a user's login and startup of a window manager and desktop as a session. When the user quits the desktop and logs out, the session ends. When another user logs in, a new session starts. The X Window System never shuts down; only desktop or window manager programs shut down. Session menus on the display manager login window list different kinds of sessions you can start—in other words, different kinds of window managers or desktops. For each session, the **Xsession** script is the startup script used to configure a user's X Window System display and to execute the selected desktop or window manager. Although this script is unnecessary for gdm, it is still used in the gdm Red Hat implementation.

Xsession is the display manager session startup script used by the Red Hat implementation of gdm (other display managers such as kdm and xdm also use **Xsession**). It contains many of the X commands also used with the **xinitrc** startup script. **Xsession** usually executes the same **xmodmap** and **xrdb** commands using the **.Xmodmap** and **.Xrsources** files in the **/etc/X11/xinit** directory. Shown here is the **Xsession** script used by gdm on Red Hat systems, which is located in the **/etc/X11/xdm** directory. Notice that any errors are saved in the user's **.xsession-errors** file in their home directory. Xsession will also read any shell scripts located in the **/etc/X11/xinit/xinitrc.d** directory. Currently this holds an input script to detect the kind of language a keyboard uses.

/etc/X11/xdm/Xsession

```
#!/bin/bash -login
# Copyright (c) 1999, 2000 Red Hat, Inc.

# redirect errors to a file in user's home directory if we can
for errfile in "$HOME/.xsession-errors" "${TMPDIR-/tmp}/xses-$USER"
"/tmp/xses-$USER"
do
          if cp /dev/null "$errfile" 2> /dev/null ;
          then
                    chmod 600 "$errfile"
                    exec > "$errfile" 2>&1
                    break
          fi
done

xsetroot -solid '#356390'

userresources=$HOME/.Xresources
usermodmap=$HOME/.Xmodmap
userxkbmap=$HOME/.Xkbmap

sysresources=/etc/X11/Xresources
sysmodmap=/etc/X11/Xmodmap
sysxkbmap=/etc/X11/Xkbmap

# merge in defaults
if [ -f "$sysresources" ]; then
           xrdb -merge "$sysresources"
fi

if [ -f "$userresources" ]; then
           xrdb -merge "$userresources"
fi

# merge in keymaps
if [ -f "$sysxkbmap" ]; then
           setxkbmap `cat "$sysxkbmap"`
           XKB_IN_USE=yes
fi

# run all system xinitrc shell scripts.
for i in /etc/X11/xinit/xinitrc.d/* ; do
           if [ -x "$i" ]; then
                     . "$i"
           fi
done

# now, we see if xdm/gdm/kdm has asked for a specific environment
case $# in
          1)
           case $1 in
                     failsafe)
                               exec xterm -geometry 80x24-0-0
                               ;;
                     gnome)
                               exec gnome-session
                               ;;
                     kde|kde1|kde2)
                               exec /usr/share/apps/switchdesk/Xclients.kde
                               ;;
                     twm)
                               # fall back to twm
                               exec /usr/share/apps/switchdesk/Xclients.twm
                               ;;
                     esac
          esac

# otherwise, take default action
if [ -x "$HOME/.xsession" ]; then
           exec "$HOME/.xsession"
elif [ -x "$HOME/.Xclients" ]; then
           exec "$HOME/.Xclients"
elif [ -x /etc/X11/xinit/Xclients ]; then
           exec /etc/X11/xinit/Xclients
else
 # should never get here; failsafe fallback
           exec xsm
fi
```

Xsession is usually invoked with an argument indicating the kind of environment to run, such as Gnome, KDE, or a window manager like Window Maker. The option for Gnome would be gnome and for KDE it would be kde.

```
Xsession gnome
```

These environments are listed in the **Xsession** script within the case statement. Here you will find entries for Gnome, KDE, and the bare-boned twm window manager. On Red Hat, gnome is invoked directly with the **gnome-session** command, whereas KDE and other window managers are invoked through scripts set up for the Red Hat switchdesk tool. These scripts contain a simple command to invoke a window manager. The scripts are held in the **/usr/share/apps/switchdesk** directory. Each file has the prefix "Xclients" and the suffix with the name of the window manger or desktop. For example, the KDE script is **Xclients.kde**, and the Window Maker script is **Xclients.windowmaker**.

If **Xsession** is not invoked with a specific environment, the user's home directory is checked for a **.xsession** or **.Xclients** script. If those scripts are missing, the system **Xclients** script is used, **/etc/X11/xinit/Xclients**.

If users want to set up their own startup files, they can copy the **Xsession** file to the their home directory and name it **.xsession** and then edit it. The following example shows a simplified **Xsession** script that executes the user's **.xsession** script if it exists. The user's **.xsession** script is expected to start a window manager or desktop.

```
#
# Xsession
#
# This is the program that is run as the client
# for the display manager.


startup=$HOME/.xsession
resources=$HOME/.Xresources
```

```
if [ -f "$startup" ]; then
           exec "$startup"
         else
           if [ -f "$resources" ]; then
                xrdb -load "$resources"
           fi
           fvwn2 &
           exec xterm -geometry 80x24+10+10 -ls
       fi
```

The X Display Manager (xdm)

The X Display Manager (xdm) manages a collection of X displays either on the local system or remote servers. xdm's design is based on the X Consortium standard X Display Manager Control Protocol (XDMCP). The xdm program manages user logins, providing authentication and starting sessions. For character-based logins, a session is the lifetime of the user shell that is started up when the user logs in from the command line interface. For xdm and other display managers, the session is determined by the session manager. The session is usually the duration of a window manager or desktop. When the desktop or window manager terminates, so does the session.

The xdm program displays a login window with boxes for a login name and password. The user logs in, and a window manager or desktop starts up. When the user quits the window manager, the X Window System restarts automatically, displaying the login window again. Authentications to control access for particular users are kept in their **.Xauthority** file.

The xdm configuration files are located in the **/usr/X11R6/lib/X11/xdm/** directory, although on Red Hat this is a link to the **/etc/X11/xdm** directory. The main xdm configuration file is **xdm-config**. Files such as **Xresources** configure how the dialog box is displayed, and **Xsetup** enables you to specify a root-window image or other windows to display. You can use the **xbanner** program to choose a graphic to display with the login dialog box. When the user starts up a session, the **Xsession** script is run to configure the user's X Window System and execute the user's window manager or desktop. This script usually calls the **.xsession** script in the user's home directory, if there is one (though this is not currently the case for Red Hat **Xsession** scripts). It holds any specific user X commands.

If you want to start xdm from the command line interface, you can enter it with the **-nodaemon** option. CTRL-C then shuts down xdm:

```
xdm -nodaemon
```

Table 35-7 lists the configuration files and directories associated with xdm. **xdm-errors** will contain error messages from xdm and the scripts it runs such as **Xsession** and **Xstartup**. Check this file if you are having any trouble with xdm.

Table 35-7: The xdm Configuration Files and Directories

Filename	**Description**
/usr/X11R6/lib/X11/xdm	xdm configuration directory; on Red Hat, this is **/etc/X11/xdm**
xdm-config	xdm configuration file

Table 35-7: The xdm Configuration Files and Directories

Filename	Description
Xsession	Startup script for user session
Xresource	Resource features for xdm login window
Xsetup	Sets up the login window and xdm login screen
Xstartup	Session startup script
xdm-errors	Errors from xdm sessions
.xsession	User's session script in the home directory; usually executed by **Xsession**
Xreset	Resets the X Window System after a session ends
.Xauthority	User authorization file where xdm stores keys for clients to read

The Gnome Display Manager

The Gnome Display Manager (gdm) manages user login and GUI interface sessions. gdm can service several displays and generates a process for each. The main gdm process listens for XDMCP requests from remote displays and monitors the local display sessions. gdm displays a login window with boxes for entering a login name and password, and also displays a pop-up menu labeled Options with entries for sessions and shutdown submenus. The sessions menu displays different window managers and desktops you can start up. On Red Hat, you can find entries for Gnome, kde, and failsafe. You can easily add entries to this menu by adding files for them in the gdm configuration directory, **/etc/X11/gdm/Sessions**.

The gdm configuration files are located in the **/etc/X11/gdm** directory. Its main configuration file is **gdm.conf**, where you can set various options, such as the logo image and welcome text to display. The **gdm** directory also contains four directories: **Init**, **Sessions**, **PostSession**, and **PreSession**. You can easily configure gdm by placing or editing files in these different directories. The **Init** directory contains scripts that are executed when gdm starts up. On Red Hat, this directory contains a **Default** script that holds X commands, such as setting the background. These are applied to the screen showing the gdm login window.

The **Sessions** directory holds session scripts. These become entries in the session menu displayed on the gdm login window options menu. For example, you could have a script called **kde** that contains the command **startkde** to run the KDE desktop. The term "kde" appears in the gdm session menu. Selecting kde executes this script and starts KDE. Currently on Red Hat, these scripts contain calls to the **/etc/X11/xdm/Xsession** script, using the filename as its argument. In the **Xsession** script, this name is used to start that particular kind of session. For example, when you select kde, the term "kde" is passed to the **Xsession** script, which then uses it to execute the **kde** command to start KDE. For example, the **Gnome** script consists of only the following lines. The term "gnome" is passed to the **Xsession** script, which then uses it to execute the **gnome-session** command to start Gnome. This design has the advantage of not having to repeat any X configuration commands, such as **xmodmap**.

```
#!/bin/bash
exec /etc/X11/xdm/Xsession gnome
```

The **PreSession** directory holds any presession commands to execute, while the **PostSession** directory holds scripts for commands you want executed whenever a session ends. Neither the **Init**, **PreSession**, or **PostSession** scripts are necessary (Red Hat currently does not include **PreSession** or **PostSession** scripts).

For gdm, the login window is generated by a program called *the greeter*. Initially, the greeter looks for icons for every user on the system, located in the **.gnome/photo** file in users' home directories. Clicking the icon automatically displays the name of the user in the login box. The user can then enter the password and click the Login button to log in.

Table 35-8 lists the configuration files and directories associated with gdm.

Table 35-8: The gdm Configuration Files and Directories

Filename	Description
/etc/X11/gdm	gdm configuration directory
gdm.conf	gdm configuration file
Init	Startup scripts for configuring gdm display
Sessions	Holds session scripts whose names appear in session menu
PreSession	Scripts execute at start of session
PostSession	Scripts execute when session ends

Starting Desktops and Window Managers

As noted in Chapter 4, the X Window System is started either automatically using a display manager with a login window or from the command line by entering the **startx** command. Your X Window System server then loads, followed immediately by the window manager. You exit the window manager by choosing an exit or quit entry in the desktop workspace menu. The display manager and some window managers, such as FVWM2 and Window Maker, give you the option of starting other window managers. If you get into trouble and the window manager hangs, you can forcibly exit the X Window System with the keys CTRL-ALT-BACKSPACE.

The desktop or window manager you start is the default window manager set up by your Linux distribution when you installed your system. Many distributions now use either Gnome or KDE as their default. For Gnome and the K Desktop, different window managers are used: kwm for the K Desktop and sawfish for Gnome. You can run Gnome or KDE applications on most window managers. To have Gnome use a particular window manager, you need to select it using the Gnome Control Center. You can also use a window manager in place of kwm for KDE. Check the window manager's Web site for current information on Gnome and KDE compatibility. Currently, Enlightenment is fully Gnome compliant, and AfterStep and Window Maker are nearly so. Red Hat only provides Enlightenment on its Publisher's Edition, and Window Maker is added for its Standard Edition.

To use a different window manager, first install it. RPM packages install with a default configuration for the window manager. If you are installing from source code you compiled, follow the included installation instructions. You can then configure Gnome or KDE to use that window manager, provided it is compliant with them. See Chapters 8 and 9 on how to configure the KDE and Gnome desktops.

You can also configure your system to start a particular window manager without either desktop. There are several ways to do this, depending on whether you are starting your window manager from the graphical login (runlevel 5) or the command line with **startx** (runlevel 3). For the graphical login, you will need to make entries in the gdm and **Xsession** files. For a command line startup, you have the options of using the Red Hat desktop switcher to select a window manager, placing a Window Manager entry in the **.wm_style** file, or creating your own **.Xclients** file that invokes one.

Gnome Display Managers and Xsession

To invoke a window manager using a display manager like gdm, you first have to create an entry for it in the login window's session menu. You then must edit the session startup script and add code to select and start up that window manager. The startup script used for the Red Hat implementation of gdm is **/etc/X11/xdm/Xsession**.

First, create an entry for the window manager in the session menu displayed in the gdm login window's options menu. Here you find files for other items listed in this menu. Simply create a file with the name of the entry you want displayed in the **/etc/X11/gdm/Sessions** directory (you can copy one of the scripts already there). On Red Hat, these scripts invoke the **/etc/X11/xdm/Xsession** script with an option for the particular window manager or desktop chosen. For example, to create an entry for Window Maker, you can create a file called **winmaker** as shown here.

/etc/X11/gdm/Sessions/winmaker

```
#!/bin/bash
exec /etc/X11/xdm/Xsession wmaker
```

To add another entry for Enlightenment, you can create a file called **enlightenment** as shown here.

/etc/X11/gdm/Sessions/enlightenment

```
#!/bin/bash
exec /etc/X11/xdm/Xsession enlightenment
```

You then must edit the **/etc/X11/xdm/Xsession** file to insert the code for selecting and starting the new window manager. **Xsession** contains detailed code, most of which you do not need to touch. In it is a case statement that lists the different window managers and desktops. Here you can enter a case entry for **wmaker** that the "wmaker" argument to Xsession will match on. The entry will execute the Window Maker program, **wmaker**. For Enlightenment you would add an entry for **enlightenment**.

You can execute the window manager program directly, as Gnome is, or you can invoke either the **switchdesk** script for the window manager or, for certain older window managers, the **RunWM** script. Enlightenment has its own **switchdesk** script, while Window Maker has both a **switchdesk** script and **RunWM** support. **RunWM** currently only supports Window Maker, LessTiff, and AfterStep. For those, it is preferable to use **RunWM** as that script will run certain checks tailored for those window managers. In the following example, Window Maker is invoked with the **RunWM** script and the **--maker** option, whereas Enlightenment is invoked with its switchdesk script, **Xclients.enlightenment**:

```
# now, we see if xdm/gdm/kdm has asked for a specific environment
case $# in
       1)
        case $1 in
               failsafe)
                      exec xterm -geometry 80x24-0-0
                      ;;
               gnome)
                      exec gnome-session
                      ;;
               kde|kde1|kde2)
                      exec /usr/share/apps/switchdesk/Xclients.kde
                      ;;
               enlightenment)
                      exec
/usr/share/apps/switchdesk/Xclients.enlightenment
                      ;;
               wmaker)
                      exec /usr/X11R6/bin/RunWM --wmaker
                      ;;
               twm)
                      # fall back to twm
                      exec /usr/share/apps/switchdesk/Xclients.twm
                      ;;
               esac
       esac
```

switchdesk

The switchdesk tool is designed to work with the X Window System started up with **startx** from the command line (runlevel 3). Switchdesk will display a list of possible window managers, and you can check the one you want to run. You then log out to shut down your current window manager. When you start up X again with **startx**, the new window manager you selected will be used. Gnome provides an entry in its system menu for the Desktop Switcher to run switchdesk. On other window managers, you may have to invoke switchdesk from a terminal window.

To add a new window manger to the list that switchdesk will display, you need to create a file for it in the **/usr/share/apps/switchdesk** directory. As previously noted, these files have the prefix "Xclients" and the suffix of the name of the window manager, like **Xclients.fvwm** for the fvwm2 window manager. The Window Maker file is shown here.

Xclients.windowmaker

```
#!/bin/bash
# (c) 2000 Red Hat, Inc.
```

```
exec /usr/bin/wmaker
```

RunWM and .wm_style

Red Hat uses a rather complex startup procedure designed to configure window managers automatically with the complete set of Red Hat menus for applications installed by its distribution. Support exists for FVWM2, LessTiff, Window Maker, and AfterStep. Red Hat's global **Xclients** file searches for a file called **.wm_style** in a user's home directory. This is where the name of your preferred window manager is placed if you select an alternative from the FVWM2 or AfterStep menus. You can also manually edit this file with a text editor and type in one. The **.wm_style** file holds a single name, such as FVWM2, AfterStep, Window Maker, or LessTiff. **Xclients** then calls the **/usr/X11R6/bin/RunWM** script with an option for the window manager to start. The **RunWM** script checks to see if the window manager is installed, and then starts it up with helpful options. As the system administrator, you can edit this file to add new entries if you want.

startx and .Xclients

You also have the option of creating your own **.Xclients** file in which you can invoke the window manager you wish. Red Hat enables users to specify their own X clients, such as window managers and desktops, in a startup file called **.Xclients**. The system X Window System startup file is called **/etc/X11/xinit/xinitrc**. For the **startx** command**,** users can also set up their own **.Xclients** file in their home directories in place of the system's **Xclients** file. To create your **.Xclients** file, you can copy the one from the system or just create the file yourself.

The invocation of the window manager is always the last command in the **.Xclients** script. The X Window System exits after finishing the execution of whatever the last command in the **.xinitrc** script is. This is an invocation of the **.Xclients** or **/etc/X11/ xinit/Xclients** script. By making the window manager the last command, exiting the window manager shuts down your X Window session. Any other programs you want to start up initially should be placed before the window manager command. You must place the command to start the window manager you want at the end of your **.Xclients** file. It is best to use the full pathname of a window manager program. These pathnames are usually located in the **/usr/X11R6/bin** directory or in the **/usr/bin** directory. The following example runs Window Maker:

```
exec /usr/X11R6/bin/wmaker
```

If you also want to load other programs automatically, such as a file manager, you can place their commands before the command for the window manager. Put an ampersand (&) after the command. The following example starts the Xterm window when the Window Maker window manager starts up:

```
xterm &
exec /usr/X11R6/bin/wmaker
```

If you are planning extensive changes, making them a few at a time is advisable, testing as you go. A simple **.Xclients** file follows, starting up a clock, terminal window, and Netscape, using the Window Maker window manager. fvwm2 is used as an alternative if Window Maker should fail to start.

.Xclients

```
#!/bin/sh

xclock -geometry 100x100-5+5 &
xterm -geometry 80x50-50+150 &
netscape &

# if the window maker is installed, run it.
if [ -f /usr/X11R6/bin/wmaker ]; then
                 exec /usr/X11R6/bin/wmaker
   else
                 exec fvwm2
  fi
```

Compiling X Window System Applications

To compile X Window System applications, you should first make sure the XFree86 development package is installed, along with any other development package you may need. These contain header files and libraries used by X Window System programs. The names of such packages contain the term **devel**—for example, **XFree86-devel**. Also, many X Window applications may need special shared libraries. For example, some applications may need the **xforms** library or the **qt** library. Gnome applications require the Gnome development libraries, while KDE applications require the KDE development libraries. You may have to obtain some of these from online sites, though most are available as part of the standard Red Hat installation.

Many X Window System applications use configure scripts that automatically detect your system's configuration and generate a **Makefile**, which can then be used to compile and install the program. An application's configure script is located in its source code directory. Simply change to that directory and execute the **configure** command. Be sure to include a preceding **./** to specify the configure script in that directory. Afterward, the command **make** compiles the program, and **make install** installs the program on your system. Check an application's **readme** and **install** files for any special instructions.

```
./configure
make
make install
```

For older applications that do not have configure scripts, use the **xmkmf** command. A **Makefile** that is configured to your system must be generated. This is done using an **Imakefile** provided with the application source code. The **xmkmf** command installed on your system can take an **Imakefile** and generate the appropriate **Makefile**. Once you have the **Makefile**, you can use the **make** command to compile the application. The **xmkmf** command actually uses a program called imake to generate the **Makefile** from the **Imakefile;** however, you should never use imake directly. Consult the Man pages for **xmkmf** and **make** for more details.

Part VII: Network Administration

Chapter List

Chapter 36: Configuring Network Connections

Overview

Most distributions enable you to configure your network during installation. If you did so, then your system is ready to go. If you need to change your configuration later, you may find the information in this chapter helpful. Administering and configuring a TCP/IP network on your Linux system is not particularly complicated. Your system uses a set of configuration files to set up and maintain your network. Table 39-2 in Chapter 39 provides a complete listing.

Instead of manually editing configuration files, you can use the GUI or cursor-based configuration tools included in many distributions that prompt you for network information. Many distributions incorporate this kind of network configuration into the installation process. If you chose not to configure your network during configuration or you need to make changes to it, you may find it easier and safer to use a network configuration tool. On Red Hat, you can use either Linuxconf or netcfg to configure your network.

Many networks now provide a service that automatically configures a system's network interface. They use a protocol called DHCP (Dynamic Host Configuration Protocol). If your network is configuring your system with DHCP, you will not have to configure it manually. All necessary information will be automatically entered into your network configuration files.

If your system does not have a direct hardware connection to a network, such as an Ethernet connection, and you dial into a network through a modem, you will probably have to set up a PPP connection. Several GUI tools are available for use with Linux that you can use to configure your PPP connection. These include Linuxconf, netcfg, and rp3. You can even initiate PPP connections from the command line.

Note If you just want to dial into a remote network using a simple command line interface, you can use a dial-in terminal program like minicom.

Network Startup Script

On Red Hat, your network interface is started up using the **network** script in the **/etc/rc.d/init.d** directory. You can manually shut down and restart your network interface by invoking this script with the **service** command and the **start**, **stop**, or **restart** options. The following command shuts down and then start up your network interface:

```
service network restart
```

To test if your interface is working, use the **ping** command with an IP address of a system on your network, such as your gateway machine. The **ping** command continually repeats until you stop it with a CTRL-C. For example, if you have a host on your network with the IP address 192.168.1.42, you could enter

```
ping 192.168.1.42
```

Hardware Specifications

In addition to your configuration files, you may also have to configure support for your networking hardware, such as Ethernet cards and modems. Ethernet cards use different modules. During installation, kmod automatically detects your Ethernet card type and has the appropriate module loaded whenever you boot up, as noted in Chapters 33 and 34. If you change your Ethernet card, it will be detected and the appropriate module loaded. An alias entry for it is entered in the **/etc/modules.conf** file, assigning it an **eth** device name such as **eth0** or **eth1** (if it is a second card). If your card is not correctly detected or if it needs certain parameters set, you will have to make an alias entry for it manually in the **/etc/modules.conf** file.

If you are using a modem, you should make sure a link by the name of **/dev/modem** exists to your modem device, which is usually one of the **/dev/ttyS***num* devices, where *num* is in the range of 0–3. For example, a modem on the second serial port has a device name of **/dev/ttyS1**, and **/dev/modem** is a link to the **/dev/ttyS1** device file (see Chapter 33). On most distributions, this is usually done for you during installation. On Red Hat, you can use a modem configuration tool called *modemtool* to create this link for you. Many modem programs and PPP configuration programs look for the **/dev/modem** file by default.

Dynamic Host Configuration Protocol (DHCP)

The Dynamic Host Configuration Protocol (DHCP) provides configuration information to systems connected to a TCP/IP network, whether the Internet or an intranet. The machines on the network operate as DHCP clients, obtaining their network configuration information from a DHCP server on their network. A machine on the network runs a DHCP client daemon that automatically downloads its network configuration information from its network's DHCP server. The information includes its IP address, along with the network's name server, gateway, and proxy addresses, including the netmask. Nothing has to be entered manually on the local system. This has the added advantage of centralizing control over network configuration for the different systems on the network. A network administrator can manage the network configurations for all the systems on the network from the DHCP server.

Note DHCP is based on the BOOTP protocol developed for diskless workstations. Check DHCP documentation for options specific to machines designed to work with BOOTP.

A DHCP server also supports several methods for IP address allocation: automatic, dynamic, and manual. Automatic allocation assigns a permanent IP address for a host. Manual allocation assigns an IP address designated by the network administrator. With dynamic allocation, a DHCP server can allocate an IP address to a host on the network only when the host actually needs to use it. Dynamic allocation takes addresses from a pool of IP addresses that hosts can use when needed and release when they are finished.

A variety of DHCP servers and clients is available for different operating systems. For Linux, you can obtain DHCP software from the Internet Software Consortium (ISC) at www.isc.org. The software package includes a DHCP server, client, and relay agent. Linux includes a DHCP server and client. The DHCP client is called **dhcpcd**, and the server is called **dhcpd**. The network information a DHCP client downloads is kept in its own network configuration files in the **/etc/dhcpc** directory. For example, here you can find a **resolv.conf** file for your network's name servers.

Configuring DHCP Client Hosts

Configuring hosts to use a DHCP server is a simple matter of setting options for the host's network interface device such as an Ethernet card. For a Red Hat Linux host, you can use netcfg to set the host to automatically access a DHCP server for network information. On the Interface panel, click the network interface device for the host, such as **eth0** for the first Ethernet card. Then click Edit. From the pop-up menu labeled Interface Configuration Protocol, select DHCP. This will set the BOOTPROTO entry in that interface's network script in the **/etc/sysconfig/network-scripts** directory, such as **ifcfg-eth0** for the first Ethernet card. You could also manually make this entry.

```
BOOTPROTO=dhcp
```

Be sure to restart your network devices with the network script to have the changes take effect.

On a Windows client, locate the TCP/IP entry for your network interface card, then open its properties window. Click the box labeled "Obtain IP address automatically". Then locate the Wins panel (usually by clicking the Advanced button) and select DHCP as the protocol you want to use.

Configuring the DHCP Server

On Red Hat systems, you can stop and start the DHCP server using the **dhcpd** command in the **/etc/rc.d/init.d** directory. Use the service command with the start, restart, and stop options. The following example starts the DHCP server. Use the **stop** argument to shut it down, and **restart** will restart it.

```
service dhcpd start
```

Configuration files for the DHCP server are kept in the **/etc/dhcpd** directory. Here is kept the primary configuration file **dhcpd.conf**, where you specify parameters that define how different DHCP clients on your network are accessed by the DHCP server. Global parameters apply to all clients. You can further create declarations that reference a particular client or different groupings of clients. Within these declarations you can enter parameters that will apply only to those clients (see Table 36-1). Declarations are implemented using different statements for different kinds of client groups. For example, to define a declaration for a particular host, you use the **host** statement as shown here:

```
host 192.168.1.2 {
        parameters
        }
```

Table 36-1: DHCP Declarations, Parameters, and Options

Entries	**Description**
Declarations	
shared-network	Used to note if some subnets share the same networks.
subnet *subnet-number netmask*	References an entire subnet of addresses.
range [*dynamic-bootp*] *low-address* [*high-address*];	Provides the highest and lowest range for dynamically allocated IP addresses.
host *hostname*	References a particular host.
group	Lets you label a group of parameters and declarations, and then use the label to apply them to subnets and hosts.
allow unknown-clients; deny unknown-clients;	Do not dynamically assign addresses to unknown clients.
allow bootp; deny bootp;	Whether to respond to **bootp** queries.
allow booting; deny booting;	Whether to respond to client queries.
Parameters	
default-lease-time *time;*	Length in seconds assigned to a lease.
max-lease-time *time;*	Maximum length of lease.
hardware *hardware-type hardware-address;*	Network hardware type (Ethernet or token ring) and address.
filename "*filename*";	Name of the initial boot file.
server-name "*name*";	Name of the server from which a client is booting.
next-server *server-name;*	Server that loads the initial boot file specified in the filename.
fixed-address *address [, address ...];*	Assign a fixed address to a client.
get-lease-hostnames *flag;*	Whether to look up and use IP addresses of clients.
authoritative; not authoritative;	Denies invalid address requests.
server-identifier hostname;	Specify the server.
Options	
option subnet-mask *ip-address;*	The client's subnet mask.
option routers *ip-address [, ip-address...];*	List of router IP addresses on client's subnet.
option domain-name-servers *ip-address [, ip-address...];*	List of domain name servers used by the client.
option log-servers *ip-address [, ip-address...];*	List of log servers used by the client.
option host-name *string;*	The client's hostname.
option domain-name *string;*	The client's domain name.
option broadcast-address *ip-*	The client's broadcast address.

Table 36-1: DHCP Declarations, Parameters, and Options	
Entries	**Description**
address;	
option nis-domain *string;*	The client's Network Information Service domain.
option nis-servers *ip-address [, ip-address...];*	NIS servers the client can use.
option smtp-server *ip-address [, ip-address...];*	List of smtp servers used by the client.
option pop-server *ip-address [, ip-address...];*	List of pop servers used by the client.
option nntp-server ip-address *[, ip-address...];*	List of nntp servers used by the client.
option www-server *ip-address [, ip-address...];*	List of Web servers used by the client.

You use the **subnet** statement to inform the DHCP server of the possible IP addresses encompassed by a given subnet. The **subnet** statement includes the netmask for the subnet.

```
subnet 192.168.1.0 netmask 255.255.255.0 {
```

With the **range** statement, you specify a range of addresses that can be dynamically allocated to clients. You enter the first and last addresses in the range.

```
range 192.168.1.5 192.168.1.128 {
```

With parameters, you can specify how the server is to treat clients. For example, the **default-lease-time** parameter sets the number of seconds a lease is assigned to a client. The **filename** parameter specifies the boot file to be used by the client. The **server-name** parameter informs the client of the host from which it is booting. The **fixed-address** parameter can be used to assign a static IP address to a client. See the Man page for **dhcpd.conf** for a complete listing.

Clients can further be provided with information they may need to access network services, such as the domain name servers that client uses or the broadcast address. This information is provided by option statements as shown here:

```
option broadcast-address 192.168.1.255;
option domain-name-servers 192.168.1.1, 192.168.1.4;
option domain-name "mytrek.com";
```

A sample **dhcpd.conf** file is shown here.

/etc/dhcpd.conf

```
subnet 192.168.1.0 netmask 255.255.255.0 {
 option routers 192.168.0.1;
 option subnet-mask 255.255.255.0;

 option domain-name "mytrek.com ";
 option domain-name-servers 192.168.1.1;
```

```
range 192.168.0.5 192.168.0.128;
default-lease-time 21600;
max-lease-time 43200;

# we want the name server to appear at a fixed address
host mynameserver {
        next-server turtle.mytrek.com;
        hardware ethernet 08:00:2b:4c:29:32;
        fixed-address 192.168.1.1;
        }
```

Note With Webmin, you can configure your DHCP server easily, entering options and parameters for subnets and hosts as well as globally.

Chapter 37: NFS, NIS, and AppleTalk

Overview

Linux provides several tools for accessing files on remotes systems connected to a network. The Network File System (NFS) enables you to connect to and directly access resources such as files or devices like CD-ROMs that reside on another machine. The Network Information Service (NIS) maintains configuration files for all systems on a network. With Samba, you can connect your Windows clients on a Microsoft Windows network to services such as shared files, systems, and printers controlled by the Linux Samba server (see Chapter 38). Netatalk enables you to connect your Linux systems to an AppleTalk network, enabling you to access remote Macintosh file systems directly, as well as to access any Apple printers such as LaserWriters.

Network File Systems: NFS and /etc/exports

NFS enables you to mount a file system on a remote computer as if it were local to your own system. You can then directly access any of the files on that remote file system. This has the advantage of allowing different systems on a network to access the same files directly, without each having to keep its own copy. Only one copy would be on a remote file system, which each computer could then access. You can find out more about NFS at its Web site at **nfs.sourceforge.net**.

NFS operates over a TCP/IP network. The remote computer that holds the file system makes it available to other computers on the network. It does so by exporting the file system, which entails making entries in an NFS configuration file called **/etc/exports**, as well as by running several daemons to support access by other systems. These include **rpc.mountd, rpc.nfsd,** and **rpc.portmapper**. Access to your NFS server can be controlled by the **/etc/hosts.allow** and **/etc/hosts.deny** files. The NFS daemons are listed here.

- **rpc.nfsd** receives NFS requests from remote systems and translates them into requests for the local system.
- **rpc.mountd** performs requested mount and unmount operations.
- **rpc.portmapper** maps remote requests to the appropriate NFS daemon.
- **rpc.rquotad** provides user disk quote management.

- **rpc.statd** provides locking services when a remote host reboots.
- **rpc.lockd** handles lock recovery for systems that have gone down.

> Note It is advisable to use NFS on a local secure network only. If used over the Internet, NFS would open your system up to nonsecure access.

On Red Hat, you start up and shut down the NFS daemons using the **/etc/rc.d/init.d/nfs** script. You can access this script directly or use the **service** command as shown here:

```
service nfs start
```

To have NFS started automatically you can use **chkconfig**, the Service Configuration tool, or **ksysv** to specify the runlevels at which it will operate. The following example will have NFS start up automatically at runlevels 3 and 5:

```
chkconfig –level 35  nfs on
```

The **nfs** script will start up the portmapper, nfsd, mountd, and rquotad daemons. To enable NFS locking, you use the **nfslock** script. This will start up the statd and lockd daemons. NFS locking provides for better recovery from interrupted operations that can occur from system crashes on remote hosts.

```
service nfslock start
```

To see if NFS is actually running, you can use the **rpcinfo** command with the **–p** option as shown here. You should see entries for **mountd** and **nfs**. If not, then NFS is not running.

```
rpcinfo -p
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100024    1   udp  32768  status
    100024    1   tcp  32768  status
    100011    1   udp    647  rquotad
    100011    2   udp    647  rquotad
    100005    1   udp  32769  mountd
    100005    1   tcp  32769  mountd
    100005    2   udp  32769  mountd
    100005    2   tcp  32769  mountd
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100021    1   udp  32770  nlockmgr
    100021    3   udp  32770  nlockmgr
```

NFS Configuration: /etc/exports

An entry in the **/etc/exports** file specifies the file system to be exported and the hosts on the network that can access it. For the file system, enter its *mountpoint,* the directory to which it was mounted on the host system. This is followed by a list of hosts that can access this file system along with options to control that access. A comma-separated list of export options placed within a set of parentheses may follow each host. For example, you might want to give one host read-only access and another read and write access. If the options are preceded by an * symbol, they are applied to any host. A list of options is provided in Table 37-1. The format of an entry in the **/etc/exports** file is shown here:

```
directory-pathname   host-designation(options)
```

Table 37-1: The /etc/exports Options

General Option	**Description**
secure	Requires that request originate on secure ports, those less than 1024. This is on by default.
insecure	Turns off the **secure** option.
ro	Allows only read-only access. This is the default.
rw	Allows read-write access.
sync	Perform all writes when requested. This is the default.
async	Perform all writes when the server is ready.
no_wdelay	Perform writes immediately, not checking to see if they are related.
wdelay	Check to see if writes are related, and, if so, wait to perform them together. Can degrade performance. This is the default.
hide	Automatically hide an exported directory that is the subdirectory of another exported directory. The subdirectory has to be explicitly mounted to be accessed. Mounting the parent directory does not allow access. This is the default.
no_hide	Do not hide an exported directory that is the subdirectory of another exported directory (opposite of **hide**). Only works for single hosts and can be unreliable.
subtree_check	Check parent directories in a file system to validate an exported subdirectory. This is the default.
no_subtree_check	Do not check parent directories in a file system to validate an exported subdirectory.
insecure_locks	Do not require authentication of locking requests. Used for older NFS versions.
User ID Mapping	
all_squash	Maps all uids and gids to the anonymous user. Useful for NFS-exported public FTP directories, news spool directories, and so forth.
no_all_squash	The opposite option to **all_squash**, and is the default setting.
root_squash	Maps requests from remote root user to the anonymous uid/gid. This is the default.
no_root_squash	Turns off root squashing. Allows the root user to access as the remote root.
anonuid **anongid**	Set explicitly the uid and gid of the anonymous account used for **all_squash** and **root_squash** options. The defaults are nobody and nogroup.

You can have several host entries for the same directory, each with access to that directory:

```
directory-pathname   host(options) host(options)  host(options)
```

You have a great deal of flexibility when specifying hosts. For hosts within your domain you can just use the hostname, whereas for those outside you need to use a fully qualified domain name. You could also just use the host's IP address. Instead of just a single host, you can reference all the hosts within a specific domain, allowing access by an entire network. A simple way to do this is to use the * for the host segment, followed by the domain name for the network, such as ***.mytrek.com** for all the hosts in the **mytrek.com** network. Instead of domain names, you could use IP network addresses using a CNDR format where you specify the netmask to indicate a range of IP addresses. You can also use an NIS netgroup name to reference a collection of hosts. The NIS netgroup name is preceded by an @ sign.

```
directory       host(options)
directory       *(options)
directory       *.domain(options)
directory       192.168.1.0/255.255.255.0(options)
directory       @netgroup(options)
```

Options in **/etc/exports** operate as permissions to control access to exported directories. Read-only access is set with the **ro** option, and read-write with the **rw** option. The **snyc** and **async** options specify whether a write operation is performed immediately (**sync**) or when the server is ready to handle it (**async**). By default, write requests are checked to see if they are related, and, if so, are written together (**wdelay**). This can degrade performance. You can override this default with **no_wdelay** and have writes executed as they are requested. If two directories are exported, where one is the subdirectory of another, then the subdirectory is not accessible unless it is explicitly mounted (**hide**). In other words, mounting the parent directory does not make the subdirectory accessible. The subdirectory remains hidden until also mounted. You can overcome this restriction with the **no_hide** option (though this can cause problems with some file systems). If an exported directory is actually a subdirectory in a larger file system, its parent directories are checked to make sure that the subdirectory is the valid directory (**subtree_check**). This option works well with read-only file systems, but can cause problems for write-enabled file systems, where filenames and directories can be changed. You can cancel this check with the **no_subtree_check** option.

Along with general options, there are also options that apply to user-level access. As a security measure, the client's root user is treated as an anonymous user by the NFS server. This is known as squashing the user. In the case of the client root user, squashing prevents the client from attempting to appear as the NFS server's root user. Should you want a particular client's root user to have root-level control over the NFS server, you can specify the **no_root_squash** option. To prevent any client user from attempting to appear as a user on the NFS server, you can classify them as anonymous users (the **all_squash** option). Such anonymous users would only have access to directories and files that are part of the anonymous group.

Normally, if a user on a client system has a user account on the NFS server, that user can mount and access his or her files on the NFS server. However, NFS requires the User ID for the user be the same on both systems. If this is not the case, he or she is considered two different users. To overcome this problem, you could use an NIS service, maintaining User ID information in just one place, the NIS password file (see the following section for information on NIS).

Examples of entries in an **/etc/exports** file are shown here. Read-only access is given to all hosts to the file system mounted on the **/pub** directory, a common name used for public access. Users, however, are treated as anonymous users (**all_squash**). Read and write access is given to the **lizard.mytrek.com** computer for the file system mounted on the **/home/foodstuff** directory. The next entry would allow access by **rabbit.mytrek.com** to the NFS server's CD-ROM. The last entry allows anyone secure access to **/home/richlp**.

/etc/exports

```
/pub                     *(ro,insecure,all_squash)
/home/foodstuff          lizard.mytrek.com(rw)
/mnt/cdrom               rabbit.mytrek.com(ro)
/home/richlp             *(secure)
```

Note Instead of editing the **/etc/exports** file directly, you can use Linuxconf's Exported File Systems panel in the Server Tasks list under the Networking heading in Config. Click the Add button to add a new entry.

Each time your system starts up the NFS server (usually when the system starts up), the **/etc/exports** file will be read, and those directories specified will be exported. When a directory is exported, an entry for it is made in the **/var/lib/nfs/xtab** file. It is this file that NFS reads and uses to perform the actual exports. Entries are read from **/etc/exports** and corresponding entries made in **/var/lib/nfs/xtab**. The **xtab** file maintains the list of actual exports.

If you want to export added entries in the **/etc/exports** file immediately, without rebooting, you can use the **exportfs** command with the **–a** option. It is helpful to add the **–v** option to display the actions that NFS is taking. Use the same options to effect any changes you make to the **/etc/exports** file.

```
exportfs -a -v
```

If you later make changes to the **/etc/exports** file, you can use the **–r** option to reexport its entries. The **–r** option will resync the **/var/lib/nfs/xtab** file with the **/etc/exports** entries, removing any other exports or any with different options.

```
exportfs -r -v
```

To both export added entries and re-export changed ones, you can combine both the **–r** and **–a** options.

```
exportfs -r -a -v
```

You can also use the **exportfs** command to manually export file systems instead of using an entry for it in the **/etc/exports** file. Export entries will be added to the **/var/lib/nfs/xtab** file directly. With the **–o** option you can list various permissions, and then follow them with the host and file system to export. The host and file system are separated by a colon. For example, to manually export the **/home/myprojects** directory to **golf.mytrek.com** with the permissions **ro** and **insecure**, you would use the following:

```
exportfs -o rw,insecure golf.mytrek.com:/home/myprojects
```

You can also use **exportfs** to unexport a directory that has already been exported, either manually or by the **/etc/exports** file. Just use the **–u** option with the host and the directory exported. The entry for the export will be removed from the **/var/lib/nfs/xtab** file. The following example will unexport the **/home/foodstuff** directory that was exported to **lizard.mytrek.com**:

```
exportfs –u lizard.mytrek.com:/home/foodstuff
```

NFS Security: /etc/hosts.allow and /etc/hosts.deny

The **/etc/hosts.allow** and **/etc/hosts.deny** are used to restrict access to services provided by your server to hosts on your network or on the Internet (if accessible). For example, you can use the **hosts.allow** file to permit access by certain hosts to your FTP server. Entries in the **hosts.deny** file would explicitly deny access to certain hosts. For NFS, you can provide the same kind of security by controlling access to specific NFS daemons.

> Note You can further secure your NFS transmissions by having them operate over TCP instead of UDP. Use the **tcp** option to mount your NFS file systems (UDP is the default); however, performance does degrade for NFS when it uses TCP.

The first line of defense is to control access to the portmapper service. The portmapper tells hosts where the NFS services can be found on the system. Restricting access does not allow a remote host to even locate NFS. For a strong level of security, you should deny access to all hosts except those that are explicitly allowed. In the **hosts.deny** file, you would place the following entry, denying access to all hosts by default. ALL is a special keyword denoting all hosts.

```
portmap:ALL
```

In the **hosts.allow** file, you would then enter the hosts on your network, or any others that you would want to permit access to your NFS server. Again, you would specify the portmapper service. Then list the IP addresses of the hosts you are permitting access. You can list specific IP addresses or a network range using a netmask. The following example allows access only by hosts in the local network, 192.168.0.0, and to the host 10.0.0.43. You can separate addresses with commas.

```
 portmap: 192.168.0.0/255.255.255.0, 10.0.0.43
```

The portmapper is also used by other services like NIS. Should you close all access to the portmapper in **hosts.deny**, you will also need to allow access to NIS services in **hosts.allow**, should you be running them. These include ypbind and ypserver. In addition, you may have to add entries for remote commands like **ruptime** and **rusers**, if you are supporting them.

In addition, it is also advisable to add the same level of control for specific NFS services. In the **hosts.deny** file you would add entries for each service, as shown here:

```
mountd:ALL
rquotad:ALL
statd:ALL
lockd:ALL
```

Then, in the **hosts.allow** file, you can add entries for each service:

```
mountd: 192.168.0.0/255.255.255.0, 10.0.0.43
rquotad: 192.168.0.0/255.255.255.0, 10.0.0.43
statd: 192.168.0.0/255.255.255.0, 10.0.0.43
lockd: 192.168.0.0/255.255.255.0, 10.0.0.43
```

You can further control access using Netfilter to check transmissions from certain hosts on the ports used by NFS services. See Chapter 40 for an explanation of Netfilter. The portmapper uses port 111 and nfsd uses 2049. Netfilter is helpful if you have a private network that has an Internet connection, and you want to protect it from the Internet. Usually a specific network device, like an Ethernet card, is dedicated to the Internet connection. The following examples assume that device **eth1** is connected to the Internet. Any packets attempting access on ports 111 and 2049 are refused.

```
iptables -A INPUT -i eth1 -p 111 -j DENY
iptables -A INPUT -i eth1 -p 2049 -j DENY
```

To enable NFS for your local network, you will have to allow packet fragments. Assuming that **etho** is the device used for the local network, you could use the following example:

```
iptables -A INPUT –I eth0 -f -j ACCEPT
```

Note A root user on a remote host can try access a remote NFS server as a root user with root level permissions. The **root_squash** option (a default) will automatically change the remote root user to the nobody (anonymous) user.

Mounting NFS File Systems: NFS Clients

Once NFS makes directories available to different hosts, those hosts can then mount those directories on their own systems, and then access them. The host needs to be able to operate as NFS clients. Current Linux kernels all have NFS client capability built in. This means that any NFS client can mount a remote NFS directory that they have access to by performing a simple mount operation.

Mounting NFS Automatically: /etc/fstab

You can mount an NFS directory either by an entry in the **/etc/fstab** file or by an explicit **mount** command. You have your NFS file systems mounted automatically by placing entries for them in **/etc/fstab** file. An NFS entry in the **/etc/fstab** file has a mount type of NFS. An NFS file system name consists of the hostname of the computer it is located on, followed by the pathname of the directory where it is mounted. The two are separated by a colon. For example, **rabbit.trek.com:/home/project** specifies a file system mounted at **/home/project** on the **rabbit.trek.com** computer. The format for an NFS entry in the **/etc/fstab** file follows. Notice that the file type is **nfs**.

```
host:remote-directory    local-directory      nfs    options    0    0
```

You can also include several NFS-specific mount options with your NFS entry. You can specify the size of datagrams sent back and forth, and the amount of time your computer waits for a response from the host system. You can also specify whether a file system is to be hard-mounted or soft-mounted. For a *hard-mounted* file system, your computer continually tries to make contact if for some reason the remote system fails to respond. A *soft-mounted* file system, after a specified interval, gives up trying to make contact and issues an error message. A hard mount is the default. A system making a hard-mount attempt that continues to fail will stop responding to user input as it tries continually to achieve the mount. For this reason, soft mounts may be preferable as they will simply stop attempting a mount that continually fails. Table 37-2 and the Man pages for **mount** contain a listing of these NFS client options. They differ from the NFS server options indicated previously.

Table 37-2: NFS Options

Option	**Description**
rsize=*n*	The number of bytes NFS uses when reading files from an NFS server. The default is 1,024 bytes. A size of 8,192 can greatly improve performance.
wsize=*n*	The number of bytes NFS uses when writing files to an NFS server. The default is 1,024 bytes. A size of 8,192 can greatly improve performance.
timeo=*n*	The value in tenths of a second before sending the first retransmission after a timeout. The default value is seven-tenths of a second.
retry=*n*	The number of minutes to retry an NFS mount operation before giving up. The default is 10,000 minutes (one week).
retrans=*n*	The number of retransmissions or minor timeouts for an NFS mount operation before a major timeout (default is 3). At that time, the connection is cancelled or a "server not responding" message is displayed.
soft	Mount system using soft mount.
hard	Mount system using hard mount. This is the default.
intr	Allow NFS to interrupt the file operation and return to the calling program. The default is not to allow file operations to be interrupted.
bg	If the first mount attempt times out, continue trying the mount in the background. The default is to fail without backgrounding.
tcp	Mount the NFS file system using the TCP protocol, instead of the default UDP protocol.

An example of an NFS entry follows. The remote system is **rabbit.mytrek.com**, and the file system is mounted on **/home/projects**. This file system is to be mounted on the local system as the **/home/dylan/projects** directory. The **/home/dylan/projects** directory must already be created on the local system. The type of system is NFS and the **timeo** option specifies that the local system wait up to 20-tenths of a second (two seconds) for a response. The mount is a soft mount and can be interrupted by NFS.

```
rabbit.mytrek.com:/home/projects /home/dylan/projects  nfs
soft,intr,timeo=20
```

Note Instead of editing the **/etc/fstab** file directly, you can use Linuxconf's Access nfs volume in the File Systems list under Config. Clicking the Add button displays Volume Specification panels for Base entries, standard file options, and NFS options. In the Base tab, boxes exist for the remote server, the remote directory (Volume), and the directory on your local system where the remote directory is to be attached (Mountpoint).

Mounting NFS Manually: mount

You can also use the **mount** command with the **-t nfs** option to mount an NFS file system explicitly. To mount the previous entry explicitly, use the following command:

```
# mount -t nfs -o soft,intr,timeo=20   \
          rabbit.mytrek.com:/home/projects   /home/dylan/projects
```

You can, of course, unmount an NFS directory with the **umount** command. You can specify either the local mountpoint or the remote host and directory, as shown here:

```
umount /home/dylan/projects
umount  rabbit.mytrek.com:/home/projects
```

On Red Hat systems, you can also mount and unmount all your NFS file systems at once with the **/etc/rc.d/init.d/netfs** script, which you can invoke with the **service** command. This script reads the NFS entries in the **/etc/fstab** file, using them to mount and unmount NFS remote directories. Using the **stop** argument unmounts the file systems, and with the **start** argument, you mount them again. The **restart** argument first unmounts and then remounts the file systems.

```
service netfs stop
```

Mounting NFS on Demand: autofs

You can also mount NFS file systems using the automount service, autofs. This requires added configuration on the client's part. The autofs service will mount a file system only when you try to access it. A directory change operation (**cd**) to a specified directory will trigger the mount operation, mounting the remote file system at that time.

The autofs service is configured using a master file to list map files, which in turn lists the file systems to be mounted. The **/etc/auto.master** file is the autofs master file. The master file will list the root pathnames where file systems can be mounted along with a map file for each of those pathnames. The map file will then list a key (subdirectory), mount options, and the file systems that can be mounted in that root pathname directory. Red Hat already implements the **/auto** directory as the root pathname for file systems automatically mounted. You could add your own in the **/etc/auto.master** file along with your own map files, if you wish. You will find that the **/etc/auto.master** file contains the following entry for the **/auto** directory, listing **auto.misc** as its map file:

```
/auto    auto.misc   --timeout 60
```

Following the map file you can add options, as shown in the previous example. The **timeout** option specifies the number of seconds of inactivity to wait before trying to automatically unmount.

In the map file, you list the key, the mount options, and the file system to be mounted. The key will be the subdirectory on the local system where the file system is mounted. For example, to mount the **/home/projects** directory on the **rabbit.mytrek.com** host to the **/auto/projects** directory, you would use the following entry:

```
projects  soft,intr,timeo=20   rabbit.mytrek.com:/home/projects
```

You could also create a new entry in the master file for an NFS file system, as shown here:

```
/myprojects    auto.myprojects   --timeout 60
```

You would then create an **/etc/auto.myprojects** file and place entries in it for NFS files system mounts, like the following:

```
dylan   soft,intr,rw   rabbit.mytrek.com:/home/projects
newgame  soft,intr,ro   lizare.mytrek.com:/home/supergame
```

Note The autofs service can be used for any file system, including floppy disks and CD-ROMs. See Chapter 32.

Network Information Service: NIS

On networks supporting NFS, many resources and devices are shared by the same systems. Normally, each system would need its own configuration files for each device or resource. Changes would entail updating each system individually. However, NFS provides a special service called Network Information Services (NIS) that maintains such configuration files for the entire network. For changes, you only need to update the NIS files. NIS works for information required for most administrative tasks, such as those relating to users, network access, or devices. For example, you can maintain user and password information with an NIS service, having only to update those NIS password files.

Note NIS+ is a more advanced form of NIS that provides support for encryption and authentication. However, it is more difficult to administer.

NIS was developed by Sun Microsystems and was originally known as Sun's Yellow Pages (YP). NIS files are kept on an NIS server (NIS servers are still sometimes referred to as YP servers). Individual systems on a network use NIS clients to make requests from the NIS server. The NIS server maintains its information on special database files called *maps.* Linux versions exist for both NIS clients and servers. Linux NIS clients easily connect to any network using NIS.

Red Hat distributions contain both the Linux NIS client and server software in RPM packages that install with default configurations. The NIS client is installed as part of the initial installation on most Linux distributions. You can use the Linuxconf Network Information Service panel in the Client Tasks list under Networking to specify the remote NIS server on your network. NIS client programs are ypbind (the NIS client daemon), ypwhich, ypcat, yppoll, ypmatch, yppasswd, and ypset. Each has its own Man page with details of its use. The NIS server programs are ypserv, ypinit, yppasswdd, yppush, ypxfr, and netgroup—each also with its own Man page. A detailed NIS HOW-TO document is available in the **/usr/share/doc/HOWTO** directory.

NIS Servers

You have significant flexibility when setting up NIS servers. If you have a small network, you may need only one NIS domain, for which you would have one NIS server. For larger networks, you can divide your network into several NIS domains, each with its own server. Even if you only have one domain, you may want several NIS slave servers. For an NIS domain, you can have a master NIS server and several NIS slave servers. The slave servers can act as backups, in case the master server goes down. A slave server only contains copies of the configuration files set up on the NIS master server.

Configuring an NIS server involves several steps, listed here:

1. Define the NIS domain name that the NIS server will work for.
2. Start the ypserv daemon.
3. In the **/var/yp/Makefile** file, set any NIS server options and specify the configuration files to manage.
4. Use ypinit to create the NIS versions of the configuration files.

You first have to define an NIS domain name. On Red Hat, you can have the NIS domain defined whenever you start up your system, by defining the NIS_DOMAIN variable in the **/etc/sysconfig/network** file. To this variable, you assign the name you want to give your NIS domain. The following example defines the NIS domain called **myturtles.nis**:

```
NIS_DOMAIN=myturtles.nis
```

When first setting up the server, you may want to define your NIS domain name without having to restart your system. You can do so with the **domainname** command, as shown here:

```
domainname myturtles.nis
```

You can start the NIS server with the ypserv startup script:

```
service ypserv start
```

Instead of the **service** command, you could reference the ypserv script directly, as shown here:

```
/etc/rc.d/init.d/ypserv start
```

Then edit the **/var/yp/Makefile** file to select the configuration files that the NIS server will maintain, along with setting any NIS server options. Red Hat has already set the standard options as well as listed most commonly used configuration files.

NIS server options are listed first. The NOPUSH option will be set to true, indicating that there are no slave NIS servers. If you are setting up any slave NIS servers for this domain, you will have to set this option to no:

```
NOPUSH = true
```

The minimum user and group ids are set to 500 on Red Hat. These are set using the MINUID and MINGID variables:

```
MINUID=500
MINGID=500
```

As Red Hat uses a shadow password and shadow group files to encrypt passwords and groups, the MERGE_PASSWD and MERGE_GROUP settings will be set to true. NIS will merge shadow password information into its password file:

```
MERGE_PASSWD=true
MERGE_GROUP=true
```

The directories where NIS will find password and other configuration files are then defined using the YPSRCDIR and YPPWDIR variables. On Red Hat this is the **/etc** directory:

```
YPSRCDIR = /etc
YPPWDDIR = /etc
```

Then the configuration files that NIS could manage are listed. Here, you will find entries like PASSWD for password, GROUP for your groups, and PRINTCAP for your printers. The current entries are shown here:

```
GROUP       = $(YPPWDDIR)/group
PASSWD      = $(YPPWDDIR)/passwd
SHADOW      = $(YPPWDDIR)/shadow
GSHADOW     = $(YPPWDDIR)/gshadow
ADJUNCT     = $(YPPWDDIR)/passwd.adjunct
#ALIASES     = $(YPSRCDIR)/aliases  # aliases could be in /etc or /etc/mail
ALIASES     = /etc/aliases
ETHERS      = $(YPSRCDIR)/ethers     # ethernet addresses (for rarpd)
BOOTPARAMS  = $(YPSRCDIR)/bootparams # for booting Sun boxes (bootparamd)
HOSTS       = $(YPSRCDIR)/hosts
NETWORKS    = $(YPSRCDIR)/networks
PRINTCAP    = $(YPSRCDIR)/printcap
PROTOCOLS   = $(YPSRCDIR)/protocols
PUBLICKEYS  = $(YPSRCDIR)/publickey
RPC         = $(YPSRCDIR)/rpc
SERVICES    = $(YPSRCDIR)/services
NETGROUP    = $(YPSRCDIR)/netgroup
NETID       = $(YPSRCDIR)/netid
AMD_HOME    = $(YPSRCDIR)/amd.home
AUTO_MASTER = $(YPSRCDIR)/auto.master
AUTO_HOME   = $(YPSRCDIR)/auto.home
AUTO_LOCAL  = $(YPSRCDIR)/auto.local
TIMEZONE    = $(YPSRCDIR)/timezone
LOCALE      = $(YPSRCDIR)/locale
NETMASKS    = $(YPSRCDIR)/netmasks
```

The actual files that are shared are the network are listed in the **all:** entry, which follows the list of configuration files. Only some of the files defined are listed as shared: those listed in the first line after **all:**. The remaining lines are automatically commented out (with a preceding # sign). You can add files by removing the # sign, or moving its entry to the first line.

```
all:  passwd group hosts rpc services netid protocols mail \
      # netgrp shadow publickey networks ethers bootparams printcap \
      # amd.home auto.master auto.home auto.local passwd.adjunct \
      # timezone locale netmasks
```

Be sure not to touch the remainder of the **Makefile**.

You then enter the **ypinit** command with the **–m** option to create the NIS database consisting of the NIS configuration files. Your NIS server will be detected, and then you will be asked to enter the names of any slave NIS servers used on this NIS domain. If there are any, enter them. When you are finished, press CTRL-D. The NIS database files are then created.

```
/usr/lib/yp/ypinit –m

At this point, we have to construct a list of the hosts which will run NIS
servers.  turtle.mytrek.com is in the list of NIS server hosts.
Please continue to add the names for the other hosts, one per line.
When you are done with the list, type a <control D>.
      next host to add:  turtle.mytrek.com
      next host to add:
The current list of NIS servers looks like this:

turtle.mytrek.com

Is this correct?  [y/n: y]  y
We need some  minutes to build the databases...
Building /var/yp/myturtles.nis/ypservers...
Running /var/yp/Makefile...
gmake[1]: Entering directory '/var/yp/myturtles.nis'
Updating passwd.byname...
Updating passwd.byuid...
Updating group.byname...
Updating group.bygid...
Updating hosts.byname...
Updating hosts.byaddr...
Updating rpc.byname...
Updating rpc.bynumber...
Updating services.byname...
Updating services.byservicename...
Updating netid.byname...
Updating protocols.bynumber...
Updating protocols.byname...
Updating mail.aliases...
gmake[1]: Leaving directory '/var/yp/myturtles.nis'
```

For an NIS slave server, you would use

```
ypinit –s masterhost
```

Should you receive the following error, it most likely means that your NIS server was not running. Be sure to start ypserv before you run ypinit.

```
failed to send 'clear' to local ypserv: RPC: Program not registeredUpdating
```

If you later need to update your NIS server files, you would change to the **/var/yp** directory and issue the **make** command.

```
cd /var/yp
make
```

The **/var/yp/securenets** file enables access by hosts to your NIS server. Hosts can be referenced by network or individually. Entries consist of a subnet mask and an IP address. For example, you could give access to all the hosts in an local network with the following entry:

```
255.255.255.0  192.168.1.0
```

For individual hosts, you can use the mask 255.255.255.255 or just the term "host", as shown here:

```
host   192.168.1.4
```

Controlling how different hosts access NIS shared data is determined in **/etc/ypserv.conf**.

Netgroups

You can use NIS to set up netgroups, which allows you to create network-level groups of users. Whereas normal groups are created locally on separate hosts, an NIS netgroup can be used for network-wide services. For example, you can use NIS netgroup to control access to NFS file systems. Netgroups are defined in the **/etc/netgroup** file. Entries consist of a netgroup name followed by member identifiers consisting of three segments: the hosts, the user, and the NIS domain:

```
group    (host, user, NIs-domain) (host, user, NIS-domain) …
```

For example, in the NIS domain **myturtles.nis**, to define a group called **myprojects** that consist of the user **chris** on the host **rabbit**, and the user **george** on the host **lizard.mytrek.com**, you would use the following:

```
myprojects (rabbit, chris, myturtles.nis) \
                          (lizard.mytrek.com, george, myturtles.nis)
```

A blank segment will match on any value. The following entry includes all users on the host **rabbit**:

```
newgame (rabbit,,myturtles.ni)
```

If your use of a group does not need both a user and a host segment, you can eliminate one by using the – sign. The following example generates a netgroup consisting just of hostnames, with no usernames:

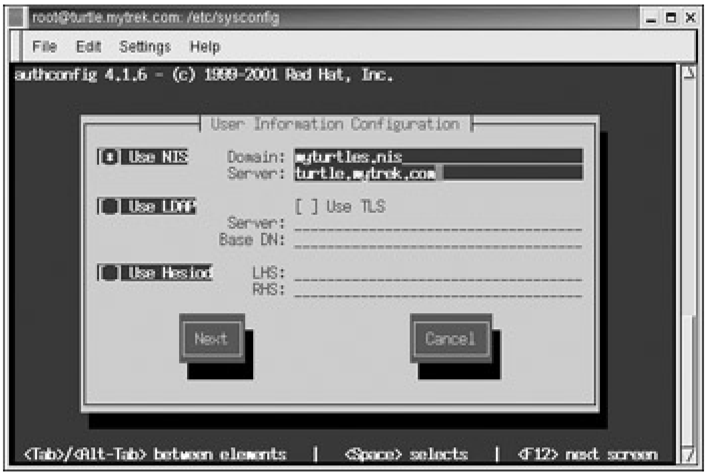
```
myservers (rabbit,-,) (turtle.mytrek.com,-,)
```

You can then reference different netgroups in various configuration files by prefixing the netgroup name with an @ sign, as shown here:

```
@newgame
```

NIS Clients

For a host to use NIS on your network, you first need to specify your NIS domain name on that host. In addition, your NIS clients need to know the name of your NIS server. If you installed Linux on a network already running NIS, you may have already entered this information during the installation process. Otherwise, on Red Hat, you can specify your NIS domain name and server with the Text Mode Setup Tool (setuptool), which you can access from the Gnome system menu or by entering the command **setup** at the shell prompt. From the menu, select Authentication. This opens the authconfig window shown in Figure 37-1. For NIS, you can enter the name of the NIS domain as well as the NIS server. Setuptool will save the NIS domain in the **/etc/sysconfig/ network** file, and the NIS server in the **/etc/yp.conf** file.



```
authconfig 4.1.6 - (c) 1999-2001 Red Hat, Inc.

User Information Configuration

[*] Use NIS       Domain: myturtles.nis
                  Server: turtle.mytrek.com

[ ] Use LDAP      [ ] Use TLS
                  Server: ________________________
                 Base DN: ________________________

[ ] Use Hesiod       LHS: ________________________
                     RHS: ________________________

<Tab>/<Alt-Tab> between elements  |  <Space> selects  |  <F12> next screen
```

Figure 37-1: Setuptool NIS domain name and server

Each NIS client host on your network then has to run the ypbind NIS client to access the server. In the client's **/etc/yp.conf** file, you need to specify the NIS server it will use. The following entry would reference the NIS server at 192.168.1.1:

```
ypserver 192.168.1.1
```

Alternatively, you can specify the NIS domain name and the server it uses:

```
domain mydomain.nis  server servername
```

Setuptool will make the following entry in **/etc/yp.conf** for the **myturtle.nis** NIS domain using the **turtle.mytrek.com** server:

```
domain myturtles.nis server turtle.mytrek.com
```

To start the NIS client, you run the ypbind script:

```
service ypbind start
```

Then, to check that all is working, you can use **ypcat** to try to list the NIS password file:

```
ypcat passwd.
```

You can use ypcat to list any of the NIS configuration files. The **ypwhich** command will display the name of the NIS server your client is using. **ypmatch** can be used to find a particular entry in a configuration file.

```
ypmatch cecelia passwd.
```

User can change their password in the NIS passwd file by using the **yppasswd** command. It works the same as the **passwd** command. You will also have to have the yppasswdd daemon running.

To ensure that the client accesses the NIS server for a particular configuration file, you should specify **nis** in the file's entry in the **/etc/nsswitch.conf** file. The **/etc/nsswitch.conf** file specifies where a host should look for certain kinds of information. For example, the following entry says to check the NIS server (**nis**) first and then the local configuration files (**files**) for passwords data:

```
passwd:   nis files
```

The **files** designation says to first use the system's own files, those on the local host. **nis** says to look up entries in the NIS files, accessing the NIS server. **nisplus** says to use NIS+ files maintained by the NIS+ server. **dns** says to perform DNS lookups and can only be used on files like **hosts** that contain hostnames. These are the entries set up by Red Hat:

```
passwd:     files nisplus nis
shadow:     files nisplus nis
group:      files nisplus nis

hosts:      files nisplus nis dns
bootparams: nisplus [NOTFOUND=return] files

ethers:     files
netmasks:   files
networks:   files
protocols:  files nisplus nis
rpc:        files
services:   files nisplus nis
netgroup:   files nisplus nis
publickey:  nisplus
automount:  files nisplus nis
aliases:    files nisplus
```

Netatalk: AppleTalk

To access Apple file systems and printers such as those on Macintosh computers, you need to use specialized servers that support the AppleTalk protocol. AppleTalk is the network protocol used for Apple Macintosh computers. AppleTalk supports file sharing and network printing, where different Macs can share each other's file systems and printers. For example, if you have a LaserWriter connected to a Macintosh, you can have other Macintosh systems access it and print on that LaserWriter. You can also access any shared file systems that may be set up on the Macintoshes on the network.

Note Mac OS X, which is based on BSD Unix, now supports NFS for file sharing.

To enable Apple systems like Macintoshes to access a Linux system, that Linux system has to emulate the AppleTalk protocols. You can do this with a Netatalk daemon. Netatalk implements the classic AppleTalk and AppleShare IP network protocol on Unix and Linux systems. AppleShare IP implements AppleTalk over an IP network. Netatalk provides support for sharing file systems, accessing printers, and routing AppleTalk. Netatalk allows a Mac machine connected to an AppleTalk network to access a Linux system as if it were an AppleTalk file and print server. Linux systems can also use Netatalk to access Mac machines connected to an AppleTalk network. The current Netatalk Web sites are **www.umich.edu/~rsug/netatalk/** and **netatalk.sourcforge.net**, with links to the FAQ and the HOW-TO sections. The Red Hat Netatalk package is part of the Powertools collection. You can download it from the Red Hat FTP site in the **powertools** directory for your release.

The Netatalk server is called afpd and is used to implement both classic AppleTalk and AppleShare IP connections. Classic AppleTalk further needs the atalkd daemon, which sets up interfaces between the kernel AppleTalk module and classic AppleTalk operations. It performs much the same function as routed and ifconfig. Several programs manage printing. The papd program lets Macs spool to a Linux printer. The pap program lets Linux systems print to an AppleTalk printer. The psf program is a PostScript printer filter for pap, and psorder enables you to print PostScript pages in reverse. The apfd program provides an interface to the Linux file system, while the nbplkup program lists all AppleTalk objects on the network. For example, **nbplkup :LaserWriter** lists the LaserWriters available. You would use the pap program to access and print to a LaserWriter. To use Linux commands to access a printer, you need to make an entry for the command in the **/etc/printcap** file and create spool, status, and lock files for it.

Netatalk requires kernel-level support for the AppleTalk Datagram Delivery Protocol (DDP) for classic AppleTalk. If your kernel does not currently support it, you either must rebuild the kernel including AppleTalk support or use a loadable module for AppleTalk, **appletalk.o**. Current kernels for most distributions include AppleTalk support. AppleShare IP only requires TCP/IP support.

Netatalk uses five configuration files, as shown in Table 37-3. The software package includes default versions you can modify. The RPM package includes the **/etc/atalk/config** file that contains documented default entries for use by the **atalk** startup script. Check the variable entries for any parameters you may want to change, such as the maximum number of allowed simultaneous users.

Table 37-3: Netatalk Configuration Files

Filename	**Description**
AppleVolumes.default	List of shared directories, including optional names
AppleVolumes.system	Maps of file extensions to Mac OS types
afpd.conf	Configuration file for **afpd** daemon that implements both classic AppleTalk and AppleShare IP (AppleTalk File system and printer daemon)
atalkd.conf	Controls the interfaces for classic AppleTalk to which Netatalk binds, enabling you to specify network numbers or zones; if empty, Netatalk detects the interfaces itself
papd.conf	Provides AppleTalk access to Linux print queues; if empty, uses **/etc/printcap**

Configuration files are automatically installed for you by the RPM package versions of Netatalk. If you are installing from the source distribution, you need to install these from default files in the source directory. Also, make sure the following lines are in your **/etc/services** file (RPM packages add these automatically):

```
rtmp     1/ddp # Routing Table Maintenance Protocol
nbp      2/ddp # Name Binding Protocol
echo     4/ddp # AppleTalk Echo Protocol
zip      6/ddp # Zone Information Protocol
afpovertcp 548/tcp # AFP over TCP
afpovertcp 548/udp
```

Netatalk is started using a startup script called **atalk**. The Red Hat RPM packages install **atalk** in the **/etc/rc.d/init.d** directory. You can also use a System V Init runlevel editor to manage startup and shutdown operations. A link is set up to start the **atalk** script when you boot. You can also use **start** and **stop** arguments directly with **atalk**:

```
 service atalk  start
```

The afpd server will implement both classic AppleTalk and AppleShare IP, though classic AppleTalk also requires that the atalkd server be running. Once apfd is started, you can then use your Macintosh client to access your shared Linux directories or printers. On your Macintosh, select AppleShare under the Chooser. Your Netatalk server should appear. For AppleShare IP, select the AppleShareIP button and enter your Netatalk server's IP address. You can then log in to the server and access shared files.

If you want to use Netatalk to mount Apple file systems locally—say, like an Apple CD-ROM disk—you need to have HFS support in your kernel. HFS (Hierarchical File System) is the Apple file system format. You can then use the **fork=netatalk** option to use Netatalk to access an Apple CD-ROM disk.

Note Currently under development is the afpfs daemon, which allows Linux systems to mount Apple file systems.

Chapter 38: Samba

Overview

With Samba, you can connect your Windows clients on a Microsoft Windows network to services such as shared files, systems, and printers controlled by the Linux Samba server. Whereas most Unix and Linux systems use the TCP/IP protocol for networking, Microsoft networking with Windows uses a different protocol, called the Server Message Block (SMB) protocol, that implements a local area network (LAN) of PCs running Windows. SMB makes use of a network interface called Network Basic Input Output System (NetBIOS) that allows Windows PCs to share resources, such as printers and disk space. One Windows PC on such a network can access part of another Windows PC's disk drive as if it were its own. SMB was originally designed for small LANs. To connect it to larger networks, including those with Unix systems, Microsoft developed the Common Internet File System (CIFS). CIFS still uses SMB and NetBIOS for Windows networking. Andrew Tridgell wrote a version of SMB he called Samba. Samba also allows Unix and Linux systems to connect to such a Windows network, as if they were Windows PCs. Unix systems can share resources on Windows systems as if they were just another Windows PC. Windows PCs can also access resources on

Unix systems as if they were Windows systems. Samba, in effect, has become a professional level, open source, and free version of CIFS. It also runs much faster than CIFS. Samba effectively enables you to use a Linux or Unix server as a network server for a group of Windows machines operating on a Windows network. You can also use it to share files on your Linux system with other Windows PCs, or to access files on a Windows PC from your Linux system, as well as between Windows PCs. On Linux systems, an **smbfs** file system enables you, in effect, to mount a remote SMB-shared directory on to your own file system. You can then access it as if it were a directory on your local system.

You can obtain extensive documentation and current releases from the Samba Web and FTP sites at **www.samba.org** and **ftp.samba.org**. RPM packages can be obtained from respective distribution FTP sites, such as **ftp.redhat.com**. Samba is also included on most Linux distributions. Other information can be obtained from the SMB newsgroup, **comp.protocols.smb**. Extensive documentation is provided with the software package and installed on your system, usually in the **/usr/share/doc** directory under a subdirectory bearing the name of the Samba release. Here, you can find extensive documentation in HTML and text format, as well as numerous examples and the current FAQs. Samba HOW-TO documentation is also available at **www.linuxdoc.org**. The examples include sample **smb.conf** files for different kinds of configuration. The home page of the SWAT configuration utility also provides Web page-based Samba documentation, as well as context-level help for different features.

The Samba software package consists of two server daemons and several utility programs: smbd, nmbd, smbclient, smbmount, smbumount, smbstatus, and testparm (see Table 38-1). One daemon, smbd, provides file and printer services to SMB clients and other systems, such as Windows, that support SMB. The nmbd utility is a daemon that provides NetBIOS name resolution and service browser support. The smbclient utility provides FTP-like access by Linux clients to Samba services. smbmount and smbumount enable Linux clients to mount and unmount Samba shared directories. The smbstatus utility displays the current status of the smb server and who is using it. You use testparm to test your Samba configuration. smbtar is a shell script that backs up SMB/CIFS-shared resources directly to a Unix tape drive. You use nmblookup to map the NetBIOS name of a Windows PC to its IP address. Included also with the package is the Samba Web administration tool (SWAT). This enables you to use a Web page interface to create and maintain your Samba configuration file, **/etc/samba/smb.conf**. Samba configuration files are kept in the **/etc/samba** directory.

Table 38-1: Samba Applications

Application	**Description**
smbd	Samba server daemon that provides file and printer services to SMD clients
nmbd	Samba daemon that provides NetBIOS name resolution and service browser support
smbclient	Provides FTP-like access by Linux clients to Samba services
smbmount	Mounts Samba share directories on Linux clients
smbumount	Unmounts Samba share directories mounted on Linux clients
smbpasswd	Changes SMB-encrypted passwords on Samba servers
smbstatus	Displays the current status of the SMB network connections

Table 38-1: Samba Applications

Application	Description
smbrun	Interface program between smbd and external programs
testparm	Tests the Samba configuration file, **smb.conf**
smbtar	Backs up SMB/CIFS-shared resources directly to a Unix tap drive
nmblookup	Maps the NetBIOS name of a Windows PC to its IP address
SWAT	Samba Web administration tool for configuring **smb.conf** with a Web browser; enables you to use a Web page interface to create and maintain your Samba configuration file, **smb.conf**

Samba provides four main services: file and printer services, authentication and authorization, name resolution, and service announcement. The SMB daemon, smbd, provides the file and printer services, as well as authentication and authorization for those services. This means users on the network can share files and printers. You can control access to these services by requiring users to provide a password. When users try to access a shared directory, they are prompted for a password. Control can be implemented in share mode or user mode. The *share* mode sets up one password for the shared resource, and then enables any user who has that password to access it. The *user* mode provides a different password for each user. Samba maintains its own password file for this purpose: **/etc/samba/smbpasswd**.

Name resolution and service announcements are handled by the nmbd server. Name resolution essentially resolves NetBIOS names with IP addresses (Microsoft plans to have this handled by DNS in the future). Service announcement, also known as *browsing*, is the way a list of services available on the network is made known to the connected Windows PCs (and Linux PCs connected through Samba).

Note For Red Hat, Samba is installed using RPM files that place SAMBA programs in **/usr/bin** and **/usr/sbin** directories. If you want to download source code or binaries in compressed archives (**.tar.gz**) from www.samba.org, the archive will extract to its own samba subdirectory. To use it, you should first uninstall the Red Hat Samba RPM files, and then extract the archive in a software directory like **/usr/local**. Be sure to add **/usr/local/samba/bin** in your PATH. Alternatively, you could copy the **samba/bin** files to **/usr/bin**, except for nmb and smbd, which should be copied to **/usr/sbin**.

Setting Up Samba

For a simple Samba setup, you should be able to use the default **smb.conf** file installed with the Linux distribution package of Samba. If you need to make changes, however, you must restart the Samba server to have the changes take effect. Starting, stopping, and restarting the Samba server is managed by the **/etc/rc.d/init.d/smb** script using the options start, stop, and restart. You can run this script with the **service** command, as shown here:

```
service smb restart
```

You could also invoke the smb script directly.

```
/etc/rc.d/init.d/smb restart
```

You can also use the desktop Service Configuration tool (serviceconf) to start and stop Samba.

To ensure name resolution, you can enter the name of your host and its IP address in the **/etc/lmhosts** file. On Windows systems, **lmhosts** entries consist of an IP address and the system's NetBIOS name, the name it is known by on a Microsoft network. For your Linux system, you can enter the IP address and the Linux system's hostname.

To test your connection from a Linux system, you can use the **smbclient** command to query the Samba server. To access the home directory of a user on the Samba server, use the IP or hostname address of the Samba server, along with the **homes** section. With the **–U** option, specify a user to connect to on the system, as shown here:

```
smbclient //turtle.mytrek.com/homes -U dylan
```

You are then prompted for a password. If the client password is different from the server password, use the server password. Once connected, you are presented with the smb client prompt as shown here. You can then access the files on the user's home directory:

```
smb: \>
```

To set up a connection for a Windows client, you need to specify the Windows workgroup name and configure the password. The workgroup name is the name that appears in the Entire Network window in the Network Neighborhood on the Windows desktop (My Network Places on Windows 2000, NT, and XP). In the **smb.conf** file, you specify the workgroup name in the **workgroup=** entry in the **global** section. The work- group name should be uppercase, no more than eight characters, and with no spaces.

You can then restart the Samba server. On a Windows client, you see the workgroup name appeared in the Entire Network folder in your Network Neighborhood. Within the workgroup is an icon for the Samba server, and within that is an icon for the user directory, as specified in the **homes** section of the **smb.conf** file.

Samba configuration options are kept in the **/etc/samba/smb.conf** file. You edit this file to make changes to the configuration. Once you finish making any changes, you should test your **smb.conf** file using the testparm program. The testparm program checks the validity of your configuration entries. By default, testparm uses the **/etc/samba/smb.conf** file, although you can supply a different configuration file as an argument:

```
testparm
```

To check your network connections, use the **smbstatus** command. This command returns a listing of all active smb connections.

Note The **/etc/samba/smbusers** file associates Windows network usernames with corresponding users on your Linux Samba server. For example, admin and administrator are made equivalent to the Linux root user.

Passwords

Connections between Windows clients and Samba servers has been further complicated by the implementation of password encryption on Microsoft networks. Current versions of Windows operating systems, including upgraded versions of Windows NT, 2000, 98, and 95, now require the use of encrypted passwords by default. Samba, on the other hand, uses unencrypted passwords by default. To enable communication between Samba servers and Windows clients you have to either change Windows clients to use unencrypted passwords, or change the Samba server to use encrypted passwords. The more secure course is to implement encrypted passwords on Samba servers, though this entails more administrative work. Note: Though not distributed by default, Samba can be built with SSL support. This SSL-enabled Samba provides support for encrypted SSL network communications. SSL-enabled Samba includes several SSL-specific configuration options, each preceded by the term **ssl**. For example, **ssl cipher** lets you determine the ciphers that can be used, and **ssl CA certFile** specifies the certificates file.

Samba also provides its own Samba password PAM module, **pam_smbpass.o**. With this module, you provide PAM authentication support for Samba passwords, enabling the use of Windows hosts on a PAM-controlled network. The module could be used for authentication and password management in your PAM samba file. The following entries in the PAM samba file would implement PAM authentication and passwords using the Samba password database:

```
auth required pam_smbpass.so nodelay
password required pam_smbpass.so nodelay
```

Be sure to enable PAM in the **smb.conf** file:

```
obey pam restrictions = yes
```

Samba Encrypted Passwords: smbpasswd

Encrypted passwords come into play if you are using user-level security instead of share-level security. With user-level security, access to Samba server resources by a Windows client is allowed only to users on that client. Each user on the Windows client has to have a corresponding user account on the Samba server. A user logs in to their Windows account and can then log in to their Samba server account. Users have to log in by providing their user name and password. Their user name and password have to be registered with the Samba server in the **/etc/samba/smbpasswd** file. You use the **smbpasswd** command to add these passwords.

To implement encrypted passwords on Samba, the Samba server then needs to maintain an encrypted version of user passwords that can be used by Windows clients. This file of encrypted passwords is **/etc/samba/smbpasswd** on Red Hat systems. Samba passwords can be added or changed for different users with the **smbpasswd** command. Initially, you will generate the Samba password file so that it will have entries for all your current Samba users. For this task, you use the **mksmbpasswd.sh** script. You input to this script the contents of the Samba server's **/etc/passwd** file, and it generates entries that can be used for encrypted passwords. You use redirection (>) to create the encrypted file. In the following example, an **/etc/samba/smbpasswd** file is initially generated on a Red Hat system by the **mksmbpasswd** script. The **cat** command with a pipe operation is used to input the contents of the **/etc/passwd** file to the **mksmbpasswd.sh** script:

```
cat /etc/passwd | mksmbpasswd.sh > /etc/samba/smbpasswd
```

If your users and their passwords are being managed by NIS, you would use the **ypcat** command to access the user passwords, as shown here:

```
ypcat passwd | mksmbpasswd.sh > /etc/samba/smbpasswd
```

You then need to change the permissions on this file to protect it from unauthorized access. The 600 option allows only read and write access by the root user:

```
chmod 600 /etc/samba/smbpasswd
```

At this point, **/etc/samba/smbpasswd** will contain entries for all your current users with dummy fields for the passwords. You then use the **smbpasswd** command to add, or later change, encrypted passwords. To add a password for a particular user, you use the **smbpasswd** command with the user's name:

```
# smbpasswd dylan
New SMB Password: new-password
Repeat New SMB Password: new-password
```

Users can use **smbpasswd** to change their own password. The following example show how you would use **smbpasswd** to change your Samba password. If the user has no Samba password, they can just press the ENTER key.

```
$ smbpasswd
Old SMB password: old-password
New SMB Password: new-password
Repeat New SMB Password: new-password
```

You also have to make sure that Samba is configured to use encrypted passwords. Set the **encrypt passwords** option to **yes** and specify the smb password file. These options are already set in the **/etc/samba/smb.conf** file (described in the following section), but they are commented with a preceding **;** symbol. Just locate the lines and remove the **;** symbols at the beginning of the lines:

```
encrypt passwords = yes
smb passwd file = /etc/samba/smbpasswd
```

You can also use SWAT to make this change. In the GLOBALS page, select Yes from the pop-up menu for the "encrypt password" entry; then save your changes by clicking the Commit Changes button.

Be sure to restart the Samba server with the following command:

```
service smb restart
```

Clear-Text Passwords on Windows Clients

If you want to use clear-text passwords for Samba, you must manually edit the Windows registry of each Windows client. This is a much riskier approach, because passwords will be transmitted across the network in clear text. On the other hand, you do not need to maintain a separate Samba password file (**/etc/samba/smbpasswd**).

For all Windows clients, you need to add an entry in the Windows registry for **EnablePlainTextPassword** and set the entry to 1:

- For Windows 95 and 98, the **EnablePlainTextPassword** entry is located in **HKEY_LOCAL_MACHINES\System\CurrentControlSet\Services\VxD\VNETSUP**. If this entry is not there, you must make one. Select Edit | New | DWORD Value from the regedit menu bar and rename the entry from New Value #1 to EnablePlainTextPassword.
- For Windows NT, you create a registry entry in **HKEY_LOCAL_MACHINE\ System\CurrentControlSet\Services\Rdr\Parameters**. Select Edit | New | DWORD Value to create the entry EnablePlainTextPassword Data: 0x01.
- For Windows 2000, you create a registry entry in **HKEY_LOCAL_MACHINE\ System\CurrentControlSet\Services\LanmanWorkStation\Parameters**. Select Edit | New | DWORD Value to create the entry EnablePlainTextPassword Data: 0x01.

Make sure Samba is not using encrypted passwords. The encrypt passwords entry in **smb.conf** needs to be commented out or turned off.

Samba Configuration: smb.conf

You configure the Samba daemon using the **smb.conf** file located in the **/etc/samba** directory. The file is separated into two basic parts: one for global options and the other for shared services. A shared service, also known as *shares,* can be either filespace services (used by clients as an extension of their native file systems) or printable services (used by clients to access print services on the host running the server). The filespace service is a directory to which clients are given access and can use the space in it as an extension of their local file system. A printable service provides access by clients to print services, such as printers managed by the Samba server.

The **/etc/samba/smb.conf** file holds the configuration for the various shared resources, as well as global options that apply to all resources. Linux installs an **smb.conf** file in your **/etc/samba** directory. The file contains default settings used for your distribution. You can edit the file to customize your configuration to your own needs. Many entries are commented with either a semicolon or a # sign, and you can remove the initial comment symbol to make them effective. Instead of editing the file directly, you may want to use the SWAT configuration utility, which provides an easy-to-use, full-screen Web page interface for entering configurations for shared resources. The SWAT configuration utility also provides extensive help features and documentation. For a complete listing of the Samba configuration parameters, check the Man page for **smb.conf**. An extensive set of sample smb.conf files is located in the **/usr/share/doc/samba*** directory in the **examples** subdirectory.

In the **smb.conf** file, global options are set first, followed by each shared resource's configuration. The basic organizing component of the **smb.conf** file is a section. Each resource has its own section that holds its service name and definitions of its attributes. Even global options are placed in a section of their own, labeled **global**. For example, each section for a filespace share consists of the directory and the access rights allowed to users of the filespace. The section of each share is labeled with the name of the shared resource. Special sections, called **printers** and **homes**, provide default descriptions for user directories and printers accessible on the Samba server. Following the special sections, sections are entered for specific services, namely access to specific directories or printers. The basic organizing component is a section. Global options are placed in a section of their own labeled **global**.

A section begins with a section label consisting of the name of the shared resource encased in brackets. Other than the special sections, the section label can be any name you want to give it. Following the section label, on separate lines, different parameters for this service are entered. The parameters define the access rights to be granted to the user of the service. For example, for a directory, you may want it to be browsable but read-only, and to use a certain printer. Parameters are entered in the format *parameter name* = *value*. You can enter a comment by placing a semicolon at the beginning of the comment line.

A simple example of a section configuration follows. The section label is encased in brackets and followed by two parameter entries. The **path** parameter specifies the directory to which access is allowed. The **writable** parameter specifies whether the user has write access to this directory and its filespace.

```
[mysection]
path = /home/chris
writable = true
```

A printer service has the same format, but requires certain other parameters. The **path** parameters specify the location of the printer spool directory. The **read-only** and **printable** parameters are set to true, indicating the service is read-only and printable. **public** indicates anyone can access it.

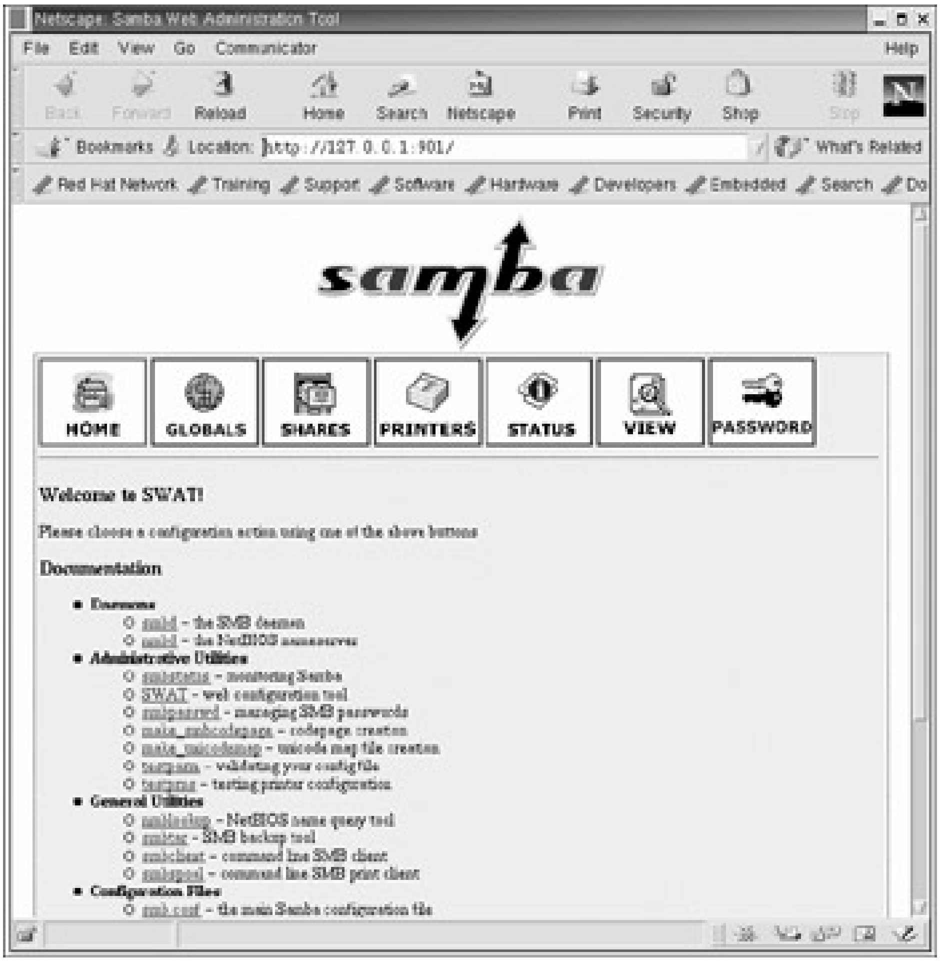
```
[myprinter]
path = /var/spool/samba
read only = true
printable = true
public = true
```

Parameter entries are often synonymous but different entries that have the same meaning. For example, **read only = no**, **writable = yes**, and **write ok = yes** all mean the same thing, providing write access to the user.

SWAT and smb.conf

SWAT is a network-based Samba configuration tool that uses a Web page interface to enable you to configure your **smb.conf** file. SWAT is, by far, the easiest and simplest way to configure your Samba server. SWAT provides a simple-to-use Web page interface with buttons, menus, and text boxes for entering values. A simple button bar across the top enables you to select the sections you want to configure. A button bar is even there to add passwords. To see the contents of the **smb.conf** file as SWAT changes it, click the View button. The initial screen (HOME) displays the index for Samba documentation (see Figure 38-1). One of SWAT's more helpful features is its context-sensitive help. For each parameter and option

SWAT displays, you can click a Help button to display a detailed explanation of the option and examples of its use.



Welcome to SWAT!

Please choose a configuration action using one of the above buttons

Documentation

- **Daemons**
  - ○ smbd - the SMB daemon
  - ○ nmbd - the NetBIOS nameserver
- **Administrative Utilities**
  - ○ smbstatus - monitoring Samba
  - ○ SWAT - web configuration tool
  - ○ smbpasswd - managing SMB passwords
  - ○ make_smbcodepage - codepage creation
  - ○ make_unicodemap - unicode map file creation
  - ○ testparm - validating your config file
  - ○ testprns - testing printer configuration
- **General Utilities**
  - ○ nmblookup - NetBIOS name query tool
  - ○ smbtar - SMB backup tool
  - ○ smbclient - command line SMB client
  - ○ smbspool - command line SMB print client
- **Configuration Files**
  - ○ smb.conf - the main Samba configuration file

Figure 38-1: SWAT home page

On Red Hat, SWAT is installed with Samba. SWAT is an **xinetd** service. As an **xinetd** service, it will be listed in the **/etc/services** and **/etc/xinetd.d/swat** files. The SWAT program uses port 901, as designated in the **/etc/services** file and shown here:

```
swat 901/tcp # Samba Web Administration Tool
```

As an **xinetd** service, SWAT will have its own **xinetd** file in the **/etc/xinetd.d** directory, **/etc/xinetd.d/swat**. SWAT is turned off by default, and its **disable** option is set to **yes**. To use SWAT, you will have to change the **disable** option to **no** as shown here:

```
# default: off
# description: SWAT is the Samba Web Admin Tool. Use swat \
#      to configure your Samba server. To use SWAT, \
#      connect to port 901 with your favorite web browser.
service swat
{
      disable = no
      port = 901
      socket_type = stream
      wait = no
      only_from = 127.0.0.1
      user = root
```

```
        server = /usr/sbin/swat
        log_on_failure += USERID
}
```

You can do this either by using chkconfig or the Service Configuration tool to turn on the SWAT service or by manually editing the **/etc/xinetd.d/swat** file and changing the **disable** option to **no**. chkconfig will edit the **/etc/xinetd.d/swat** file for you, making this change (see Chapter 22 for more information about chkconfig). The following example shows the way you would enable SWAT with the **chkconfig** command:

```
chkconfig swat on
```

With chkconfig, you will not have to manually restart the **xinetd** server. However, if you manually edit the file, you will also have to restart the server to have the change take effect. On Red Hat, you can do this simply using the **xinetd** script, as shown here:

```
service xinetd restart
```

Before you use SWAT, back up your current **smb.conf** file. SWAT overwrites the original, replacing it with a shorter and more concise version of its own. The **smb.conf** file originally installed lists an extensive number of options with detailed explanations. This is a good learning tool, with excellent examples for creating various kinds of printer and directory sections. Simply make a backup copy:

```
cp /etc/samba/smb.conf /etc/samba/smb.bk
```

On Red Hat, you can start up SWAT by selecting the Samba Configuration entry in the Gnome System menu. This will open up your Web browser to the SWAT page using the localhost IP address, 127.0.0.1 and port 901 as shown in Figure 38-1. You can also open your browser and enter the IP address 127.0.0.1 with port 901 to access SWAT.

```
http://127.0.0.1:901
```

You can start up SWAT from a remote locate by entering the address of the Samba server it is running on, along with its port (901) into a Web browser. However, you will first have to enable this feature in the **/etc/xinetd.d/swat** file. Currently the **only_from** line in this file restricts access to just localhost. To enable access from any remote system, just remove this line. If you want to provide access to certain specific hosts, you can list them after 127.0.0.1 on the **only_from** line. Be sure to restart SWAT after any changes. The following example enables access from both 127.0.0.1 and **rabbit.mytrek.com**:

```
only_from 127.0.0.1 rabbit.mytrek.com
```

The following URL entered into a Web browser on a remote system would then display the Web page interface for SWAT on the **turtle.mytrek.com** Samba server:

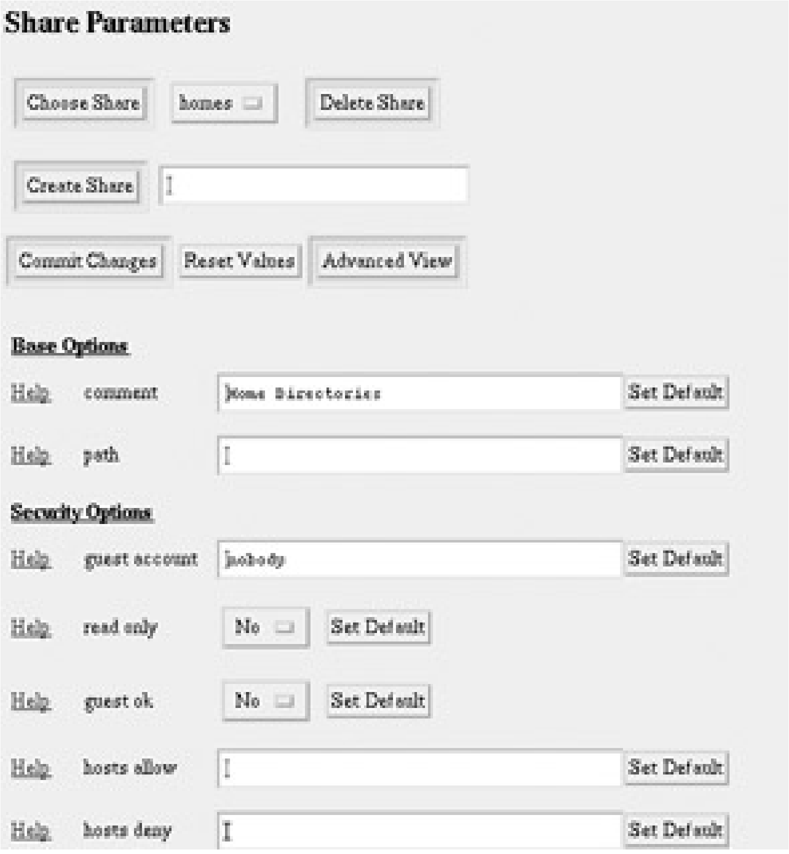
```
http://turtle.mytrek.com:901
```

You are first asked to enter a username and a password. To configure Samba, you need to enter **root** and the root password. (If you are connecting from a remote system, it is *not* advisable to enter the root password in clear text—see Chapter 31.) The main SWAT page is displayed with a button bar, with buttons for links for HOME, GLOBALS, SHARES, PRINTERS, STATUS, VIEW, and PASSWORD (see Table 38-2). You can use STATUS to list your active SMB network connections.

Table 38-2: SWAT Configuration Pages

Page	**Description**
HOME	SWAT home page listing documentation resources
GLOBALS	Configure the global section for Samba
SHARES	Select and configure directories to be shared (shares)
PRINTERS	Set up access to printers
STATUS	Check the status of the Samba server, both smbd and nmbd; list clients currently active and the actions they are performing. You can restart, stop, or start the Samba server from this page
VIEW	Display the **smb.conf** configuration file
PASSWORD	Set up password access for the server and users that have access

For the various sections, SWAT can display either a basic or advanced version. The basic version shows only those entries needed for a simple configuration, whereas the advanced version shows all the possible entries for that type of section. A button— labeled Advanced View and Basic View, respectively—is at the top of the section page for toggling between the advanced or basic versions (see Figure 38-2). Section pages for printers and shares have added buttons and a menu for selecting the particular printer or share you want to configure. The term "share," as it's used here, refers to directories you want to make available through Samba. When you click the SHARES button, you initially see only a few buttons displayed at the top of the SHARES page. You use these buttons to create new sections or to edit sections already set up for shares. For setting up a new Share section, you enter its name in the box next to the Create Share button and then click that button. The new share name appears in the drop-down menu next to the Choose Share button. Initially, this button is blank. Click it to display the list of current Share sections. Select the one you want, and then click the Choose Share button. The page then displays the entries for configuring a share. For a new share, these are either blank or default values. For example, to select the Homes section that configures the default setting for user home directories, click the drop-down menu where you find a Homes entry. Select it, and then click the Choose Share button. The entries for the Homes section are displayed. The same process works for the Printers page, where you can select either the Printers section or Create sections for particular printers.



Base Options

comment: Home Directories

path:

Security Options

guest account: nobody

read only: No

guest ok: No

hosts allow:

hosts deny:

Figure 38-2: SWAT Share page showing Homes section

Note Samba automatically creates entries for any printer already configured for use on your system or network. It reads these from your **/etc/printcap** file. You will need to edit the printer entries to control access to your printers. For Samba to use a printer, it first has to be configured on your system as either a local or network printer (see printconf in Chapters 4 and 33). Keep in mind that a network printer could be a printer connected to a Windows system.

In Figure 38-2, notice the Help links next to each entry. Such a link displays a Web page showing the Samba documentation for **smb.conf**, positioned at the appropriate entry. In this figure, the Guest OK part of the documentation is displayed after the user clicks the Help link next to the Guest OK entry.

When you finish working on a section, click the Commit Changes button on its page to save your changes. Do this for each separate page you work on, including the GLOBALS page. Clicking the Commit Changes button generates a new version of the **smb.conf** file. To have the Samba server read these changes, you then have to restart it. You can do this by clicking on the Restart smb button on the Status page.

The basic procedures for creating a new share using SWAT include the following steps:

1. Select the Share page and, in the Create Share text box, enter the name of the new share.
2. Click the Create Share button to open a configuration page for the new share. The name of the new share will appear in the pop-up menu next to the Choose Share button.

3. Enter various options. For the Basic Options, you will have to specify the directory for the share in the "path" text box. In the "comment" text box, you enter the label that will appear on Windows for the share.
4. Click the Commit Changes button to save your share entry to the Samba configuration file, **smb.conf**. Then restart the Samba server to effect your changes (click the Restart smb button on the Status page).

You can follow a similar procedure to add a new printer, but make sure the printer is also configured on the system with the Samba server.

You can, of course, edit the **/etc/samba/smb.conf** file directly. This is a simple text file you can edit with any text editor. You still must restart the smb server to have the changes take effect, which you can do manually with the following command:

```
service smb restart
```

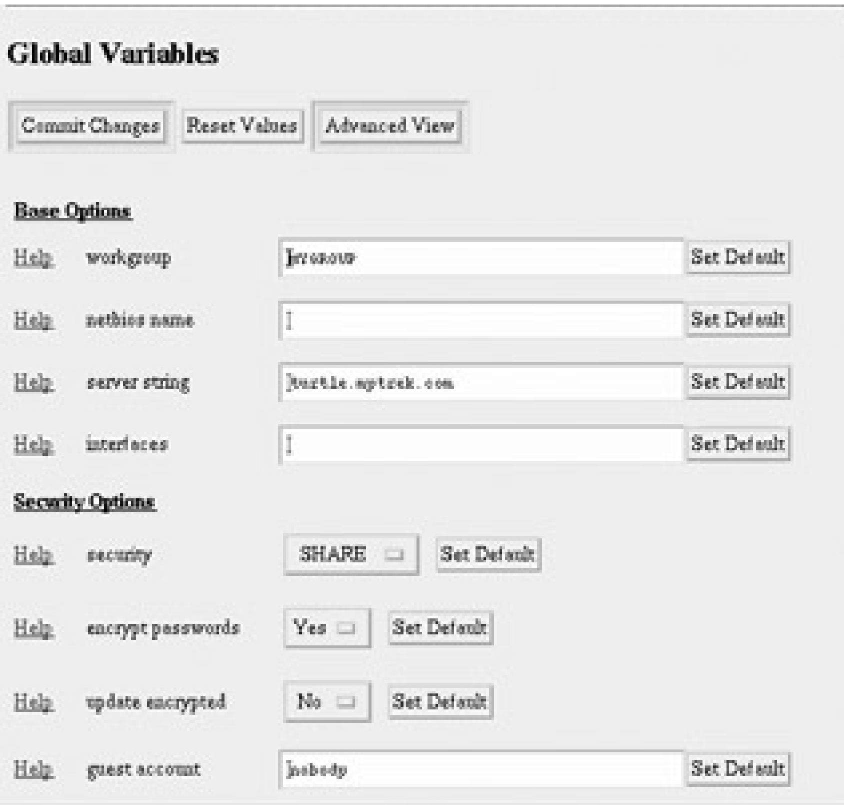
The following example shows an **smb.conf** file generated by SWAT for a simple configuration. This is much smaller than the comment-intensive versions originally installed with Samba. In this configuration, share-level security is implemented and password encryption is enabled. A share called myprojects is defined, which has guest access and is writeable. A printer share called myhp is also defined and also supports guest access.

```
# Samba config file created using SWAT
# from localhost.localdomain (127.0.0.1)
# Date: 2001/09/09 01:09:07

# Global parameters
[global]
           server string = Samba Server
           security = SHARE
           encrypt passwords = Yes
           ssl CA certFile = /usr/share/ssl/certs/ca-bundle.crt
           log file = /var/log/samba/%m.log
           max log size = 0
           socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
           dns proxy = No
           printing = lprng

[homes]
           comment = Home Directories
           path = /home
           writeable = Yes
           guest ok = Yes

[printers]
           comment = All Printers
           path = /var/spool/samba
           guest ok = Yes
           printable = Yes
           browseable = No

[myprojects]
           path = /myprojects
           writeable = Yes
           guest ok = Yes

[myhp]
```

```
            path = /var/spool/samba
            writeable = Yes
            guest ok = Yes
            printable = Yes
            printer = myhp
            oplocks = No
            share modes = No
```

Global Section

The Global section determines configuration for the entire server, as well as specifying default entries to be used in the home and directory segments. In this section, you find entries for the workgroup name, password configuration, and directory settings. Several of the more important entries are discussed here. Figure 38-3 shows the Global Variables page on the SWAT that you can use to set global options. The Basic View of this page lists the options you would most likely need.



Base Options

Help workgroup mygroup

Help netbios name

Help server string turtle.mytrek.com

Help interfaces

Security Options

Help security SHARE

Help encrypt passwords Yes

Help update encrypted No

Help guest account nobody

Figure 38-3: SWAT Global Variables page

The workgroup entry specifies the workgroup name you want to give to your network. This is the workgroup name that appears on the Windows client's Network Neighborhood window. The default workgroup entry in the **smb.conf** file is shown here:

```
[global]

# workgroup = NT-Domain-Name or Workgroup-Name
 workgroup = MYGROUP
```

The workgroup name has to be the same for each Windows client that the Samba server supports. On a Windows client, the workgroup name is usually found on the Network

Identification or General panels in the System tool located in the Control Panels window. On many clients this is defaulted to WORKGROUP. If you want to keep this name, you would have to change the workgroup entry in the **smb.conf** file accordingly. The workgroup entry and the workgroup name on each Windows client has to be the same.

```
workgroup = WORKGROUP
```

The server string entry holds the descriptive name you want displayed for the server on the client systems. On Windows systems, this is the name displayed on the Samba server icon. The default is "Samba Server", but you can change this to any name you want.

```
# server string is the equivalent of the NT Description field
 server string = Samba Server
```

Samba resources are normally accessed with either share or user level security. On a share level, any user can access the resource without having to log in to the server. On a user level, each user has to log in, using a password. Furthermore, Windows 98, ME, NT, and XP clients use encrypted passwords for the logging-in process. For these clients you will have to enable encrypted passwords. The default for encrypted passwords is no, so you will need to change it to yes. In the **smb.conf** file, the security option is set to the level you want and the encrypt passwords option is set to yes to enable encryption.

```
security = user
encrypt passwords = yes
```

If you want share-level security, specify share as the security option.

```
security = share
```

On the SWAT GLOBALS page, select the security level from the pop-up menu, either user or share. Then select Yes for the encrypt passwords entry.

As a security measure, you can restrict access to SMB services to certain specified local networks. On the host's network, type the network addresses of the local networks for which you want to permit access. The localhost (127) is always automatically included. The next example allows access to two local networks:

```
hosts allow = 192.168.1. 192.168.2.
```

To enable printing, allow Samba to load the printer descriptions from your **printcap** file. Although you can specify a particular print system type with the printing entry, this usually is unnecessary.

```
 printcap name = /etc/printcap
 load printers = yes
```

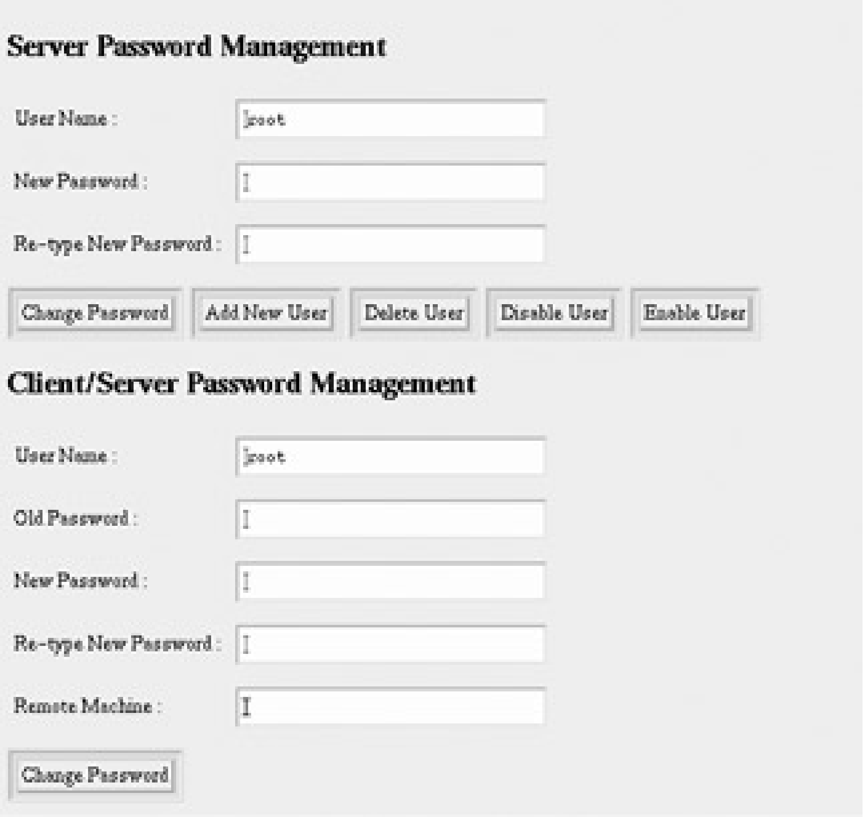
You can use a guest user to make resources available to anyone without requiring a password. A guest user login would handle any users who log in without a specific account. On Linux systems, by default Samba will use the **nobody** user as the guest user. Alternatively, you can set up and designate a specific user to use as the guest user. You designate the guest user with the "guest account" entry in the **smb.conf** file. The commented **smb.conf** file provided by Samba currently lists a commented entry for setting up a guest user called **pcguest**. You can make this a user you want used as the guest user. Be sure to add the guest user to the password file:

```
guest account = pcguest
```

On SWAT you can specify a guest account entry on the GLOBALS page. By default this is already set to the **nobody** user.

Passwords

As noted previously, user-level security requires that each user log in to the Samba server using passwords. Samba can use either clear text or encrypted passwords, although current Windows clients support encrypted passwords. You can use the **smbpasswd** command to add and change Samba passwords. On SWAT, you enable password encryption on the GLOBALS page and manage passwords on the Passwords page, as shown in Figure 38-4. On the Server Password Management section you can add, change, remove, enable, or disable users. To add a new user, enter the user name and password, then click Add New User. As the root user on the Samba server, you can add new passwords as well as enable or disable current ones. Normal users can use the Client/Server Password Management section to change their own passwords.



Server Password Management

User Name : root

New Password :

Re-type New Password :

Client/Server Password Management

User Name : root

Old Password :

New Password :

Re-type New Password :

Remote Machine :

Figure 38-4: SWAT Passwords page

Homes Section

The Homes section specifies default controls for accessing a user home directory through the SMB protocols by remote users. Setting the Browseable entry to No prevents the client from listing the files with the browser, such as that used by a file manager to display files and directories (for example, Explorer on Windows). The Writable entry specifies whether users have read and write control over files in their home directories. On SWAT, you simply select the SHARES page, select the Homes entry from the drop-down menu, and click Choose Share (see Figure 38-2).

```
[homes]
 comment = Home Directories
 browseable = no
 writable = yes
```

Printers Section

The Printers section specifies the default controls for accessing printers. These are used for printers for which no specific sections exist. In this case, Samba uses printers defined in the server's **printcap** file.

In this context, setting Browseable to No simply hides the Printers section from the client, not the printers. The path entry specifies the location of the spool directory Samba will use for printer files. To enable printing at all, the printable entry must be set to Yes. To allow guest users to print, set the Guest OK entry to Yes. The Writable entry set to No prevents any kind of write access, other than the printer's management of spool files. On SWAT, select the PRINTER page and the Printers entry in the drop-down menu, and then select Choose Printers. A standard implementation of the Printers section is shown here:

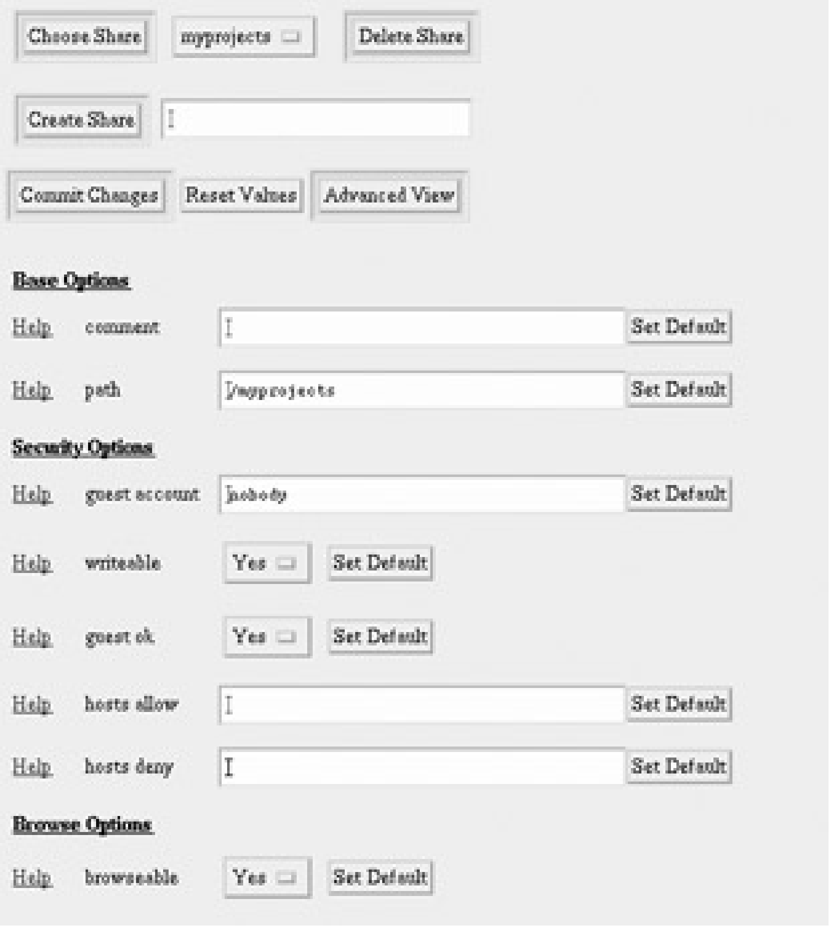
```
 [printers]
 comment = All Printers
 path = /var/spool/samba
 browseable = No
 guest ok = Yes
 writable = No
 printable = Yes
```

If you can't print, be sure to check the Default Print entry. This specifies the command the server actually uses to print documents.

Shares

Sections for specific shared resources, such as directories on your system, are usually placed after the Homes and Printers sections. For a section defining a shared directory, enter a label for the system. Then, on separate lines, enter options for its pathname and the different permissions you want to set. In the **path** = option, specify the full pathname for the directory. The **comment** = option holds the label to be given the share. You can make a directory writable, public, or read-only. You can control access to the directory with the Valid Users entry. With this entry, you can list those users permitted access. For those options not set, the defaults entered in the Global, Home, and Printer segments are used.

On SWAT, you use the SHARES page to create and edit shared directories. Select the one you want to edit from the drop-down menu and click Choose Share. The Basic View shows the commonly used entries. For entries such as Valid Users, you need to select the Advanced View. Be sure to click Commit Changes before you move on to another Share or Printers section (see Figure 38-5).



Base Options

comment

path /myprojects

Security Options

guest account nobody

writeable Yes

guest ok Yes

hosts allow

hosts deny

Browse Options

browseable Yes

Figure 38-5: SWAT Samba Share

The following example is the myprojects share generated by SWAT from the share page shown in Figure 38-5. Here the **/myprojects** directory is defined as a share resource that is open to any user with guest access.

```
[myprojects]
     comment = Great Project Ideas
     path = /myprojects
     writeable = Yes
     guest ok = Yes
     printable = Yes
```

To limit access to certain users, you can list a set of valid users. Setting the public option to No closes it off from access by others.

```
[mynewmusic]
 comment =  Service
 path = //home/specialprojects
 valid users = mark
 public = no
 writable = yes
 printable = no
```

To allow complete public access, set public entry to Yes, with no valid user's entry.

```
 [newdocs]
 path = /home/newdocs
 public = yes
 writable = yes
 printable = yes
```

To set up a directory that can be shared by more than one user, where each user has control of the files he or she creates, simply list the users in the Valid Users entry. Permissions for any created files are specified by the Create Mask entry. In this example, the permissions are set to 765, which provides read/write/execute access to owners, read/write access to members of the group, and only read/execute access to all others:

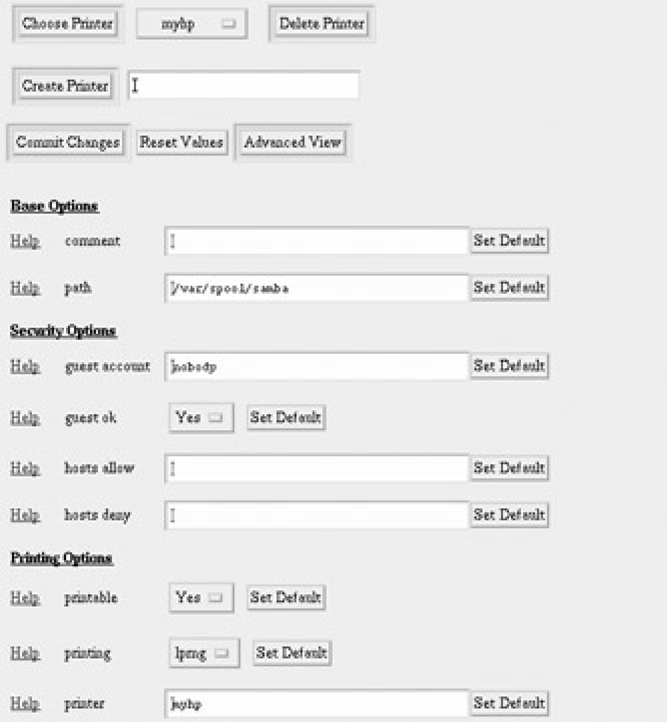
```
[myshare]
 comment = Writer's projects
 path = /usr/local/drafts
 valid users = justin chris dylan
 public = no
 writable = yes
 printable = no
 create mask = 0765
```

For more examples, check those in the original **smb.conf** file that shows a Share section for a directory **fredsdir**.

Printers

Access to specific printers is defined in the Printers section of the **smb.conf** file. You can also configure printers in the SWAT Printers page. For a printer, you need to include the printer and printable entries. With the Printers entry, you name the printer, and by setting the Printable entry to Yes, you allow it to print. You can control access to specific users with the valid users entry and by setting the public entry to No. For public access, set the public entry to Yes. On SWAT, you can create individual Printers sections on the PRINTERs page. Default entries are already set up for you.

The following example sets up a printer accessible to guest users. This opens the printer up to use by any user on the network. Users need to have write access to the printer's spool directory, located in **/var/spool/samba**. Keep in mind that any printer has to first be installed on your system. The following printer was already installed as myhp and has an **/etc/printcap** entry with that name. You can use printconf to install your printer, giving it a name and selecting it driver (see Chapters 4 and 33). The SWAT Printer page used to generate this printer entry is shown in Figure 38-6.



myhp

Base Options

comment

path /var/spool/samba

Security Options

guest account nobody

guest ok Yes

hosts allow

hosts deny

Printing Options

printable Yes

printing lprng

printer myhp

Figure 38-6: SWAT printer share

```
[myhp]

      path = /var/spool/samba
      writeable = Yes
      guest ok = Yes
      printable = Yes
      printer = myhp
      oplocks = No
      share modes = No
```

As with shares, you can restrict printer use to certain users, denying it from public access. The following example sets up a printer accessible only by the users **larisa** and **aleina** (you could add other users if you want). Users need to have write access to the printer's spool directory.

```
[larisalaser]

      path = /var/spool/samba
      writeable = Yes
      valid users = larisa aleina
      public = no
      printable = Yes
      printer = myhp
      oplocks = No
      share modes = No
```

Note Though SWAT is preferred, you can also use Linuxconf and Webmin to configure Samba.

Variable Substitutions

For string values assigned to parameters, you can incorporate substitution operators. This provides greater flexibility in designating values that may be context-dependent, such as usernames. For example, suppose a service needs to use a separate directory for each user who logs in. The path for such directories could be specified using the **%u** variable that substitutes in the name of the current user. The string **path = /tmp/%u** would become **path = /tmp/justin** for the **justin** user and **/tmp/dylan** for the **dylan** user. Table 38-3 lists several of the more common substitution variables.

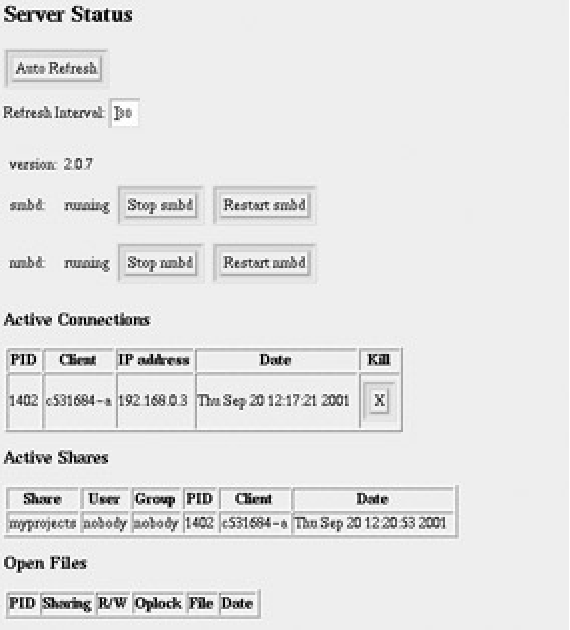
Table 38-3: Samba Substitution Variables

Variable	**Description**
%S	Name of the current service
%P	Root directory of the current service
%u	Username of the current service
%g	Primary group name of the user
%U	Session username (the username the client wanted)
%G	Primary group name of session user
%H	Home directory of the user
%v	Samba version
%h	Internet hostname on which Samba is running
%m	NetBIOS name of the client machine
%L	NetBIOS name of the server
%M	Internet name of the client machine
%N	Name of your NIS home directory server
%p	Path of the service's home directory
%d	Process ID of the current server process
%a	Architecture of the remote machine
%I	IP address of the client machine
%T	Current date and time

Testing the Samba Configuration

After you make your changes to the **smb.conf** file, you can then use the testparm program to see if the entries are correctly entered. testparm checks the syntax and validity of Samba entries. By default, testparm checks the **/etc/samba/smb.conf** file. If you are using a different file as your configuration file, you can specify it as an argument to testparm. You can also have testparm check to see if a particular host has access to the service set up by the configuration file.

With SWAT, the Status page, shown in Figure 38-7, will list your connections and shares. From the command line, you can use the **smbstatus** command to check on current Samba connections on your network.



Server Status

Refresh Interval: 30

version: 2.0.7

smbd: running

nmbd: running

Active Connections

PID	Client	IP address	Date	Kill
1402	c531684-a	192.168.0.3	Thu Sep 20 12:17:21 2001	X

Active Shares

Share	User	Group	PID	Client	Date
myprojects	nobody	nobody	1402	c531684-a	Thu Sep 20 12:20:53 2001

Open Files

PID	Sharing	R/W	Oplock	File	Date

Figure 38-7: SWAT Samba network status

To check the real-time operation of your Samba server, you can log in to a user account on the Linux system running the Samba server and connect to the server.

Domain Logons

Samba also supports domain logons whereby a user can log on to the network. Logon scripts can be set up for individual users. To configure such netlogin capability, you need to set up a netlogon share in the **smb.conf** file. The following sample is taken from the original **smb.conf** file. This share holds the **netlogon** scripts—in this case, the **/home/netlogon** directory—which should not be writable, but it should be accessible by all users (Guest OK):

```
[netlogon]
 comment = Network Logon Service
 path = /home/netlogon
 guest ok = yes
 writeable = no
 share modes = no
```

The Global section would have the following parameters enabled:

```
domain logons = yes
```

With netlogon, you can configure Samba as an authentication server for both Linux and Windows hosts. A Samba user and password needs to be set up for each host. In the Global section of the **smb.conf** file, be sure to enable encrypted passwords, user-level security, and domain logons, as well as a operating system level of 33 or more:

```
 [global]
 encrypt passwords = Yes
 security = user
 domain logons = Yes
 os level = 33
```

Note You can also configure Samba to be a Primary Domain Controller (PDC) for Windows

NT networks. As a PDC, Samba can handle domain logons, retrieve lists of users and groups, and provide user-level security.

Accessing Samba Services with Clients

Client systems connected to the SMB network can access the shared services provided by the Samba server. Windows clients should be able to access shared directories and services automatically through the Network Neighborhood and the Entire Network icons on a Windows desktop. For other Linux systems connected to the same network, Samba services can be accessed using special Samba client programs. With smbclient**,** a local Linux system can connect to a shared directory on the Samba server and transfer files, as well as run shell programs. With smbmount**,** directories on the Samba server can be mounted to local directories on the Linux client.

Note Several Samba browser clients are available for Gnome and KDE. For KDE you can use Komba2 (Red Hat RPM downloadable from **apps.kde.com**). For Gnome you can use Gnomba.

smbclient

smbclient operates like FTP to access systems using the SMB protocols. Whereas with an FTP client you can access other FTP servers or Unix systems, with smbclient you can access SMB-shared services, either on the Samba server or on Windows systems. Many smbclient commands are similar to FTP, such as **mget** to transfer a file or **del** to delete a file. The smbclient program has several options for querying a remote system, as well as connecting to it (see Table 38-4). See the smbclient Man page for a complete list of options and commands. The smbclient program takes as its argument a server name and the service you want to access on that server. A double slash precedes the server name and a single slash separates it from the service. The service can be any shared resource, such as a directory or a printer. The server name is its NetBIOS name, which may or may not be the same as its IP name. For example, to specify the **myreports** shared directory on the server named **turtle.mytrek.com**, use **//turtle.mytrek.com/myreports**. If you must specify a pathname, use backslashes for Windows files and forward slashes for Unix/Linux files:

```
//server-name/service
```

Table 38-4: smbclient Options

Option	**Description**
password	The password required to access the specified service on the server. If no password is supplied, the user is prompted to enter one.
-s *smb.conf*	Specify the pathname to **smb.conf** file.
-B *IP_address*	Specify the broadcast IP address.
-O *socket_options*	List the socket options.
-R *name resolve order*	Use these name resolution services only.
-M *host*	Send a winpopup message to the host.
-i *scope*	Use this NetBIOS scope.
-N	Don't ask for a password.

Table 38-4: smbclient Options

Option	Description
-n *netbios name*	Use this name as my NetBIOS name.
-d *debuglevel*	Set the debug level.
-P	Connect to the service as a printer.
-p *port*	Connect to the specified port.
-l *log basename*	Base name for log/debug files.
-h	Print this help message.
-I *IP_address*	Specify the IP address to connect to.
-E	Write messages to stderr instead of stdout.
-U *username*	Specify the user to log in to on the remote system.
-L *host*	List the shares available on the specified host.
-t *terminal code*	Terminal I/O code used {sjis|euc|jis7|jis8|junet|hex}.
-m *max protocol*	Set the max protocol level.
-W *workgroup*	Set the workgroup name.
-Tc|x	Command line **tar** operation.
-D *directory*	Start from this directory.
-c *command_string*	Execute semicolon-separated commands.
-b *xmit/send buffer*	Change the transmit/send buffer (default: 65520).

You can also supply the password for accessing the service. Enter it as an argument following the service name. If you do not supply the password, you are prompted to enter it.

You can then add several options, such as the remote username or the list of services available. With the **–I** option, you can specify the system using its IP address name. You use the **–U** option and a login name for the remote login name you want to use on the remote system. Attach **%** with the password if a password is required. With the **–L** option, you can obtain a list of the services provided on a server, such as shared directories or printers. The following command will list the shares available on the host **turtle.mytrek.com**:

```
smbclient -L turtle.mytrek.com
```

To access a particular directory on a remote system, enter the directory as an argument to the **smbclient** command, followed by any options. For Windows files, you use backslashes for the pathnames, and for Unix/Linux files you use forward slashes. Once connected, an **smb** prompt is displayed and you can use smbclient commands such as **get** and **put** to transfer files. The **quit** or **exit** commands quit the smbclient program. In the following example, smbclient accesses the directory **myreports** on the **turtle.mytrek.com** system, using the **dylan** login name:

```
smbclient //turtle.mytrek.com/myreports -I 192.168.0.1 –U dylan
```

In most cases, you can simply use the server name to reference the server, as shown here:

```
smbclient //turtle.mytrek.com/myreports -U dylan
```

If you are accessing the home directory of a particular account on the Samba server, you can simply specify the **homes** service. In the next example, the user accesses the home directory of the **aleina** account on the Samba server, after being prompted to enter that account's password:

```
smbclient //turtle.mytrek.com/homes -U aleina
```

You can also use **smbclient** to access shared resources located on Windows clients. Specify the computer name of the Windows client along with its shared folder. In the next example, the user accesses the windata folder on the Windows client named lizard. The folder is configured to allow access by anyone, so the user just presses the ENTER key at the password prompt.

```
$ smbclient //lizard/windata
added interface ip=192.168.0.2 bcast=192.168.0.255 nmask=255.255.255.0
Got a positive name query response from 192.168.0.3 ( 192.168.0.3 )
Password:
Domain=[WORKGROUP] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]
smb: \> ls
  .                                   D         0  Sat Sep  8 17:29:19 2001
  ..                                  D         0  Sat Sep  8 17:29:19 2001
  hi                                  A        10  Sat Sep  8 17:29:27 2001
  mynewdoc.doc                        A         0  Sat Sep  8 16:59:13 2001
           39997 blocks of size 1048576. 39930 blocks available
smb: \> mget hi
Get file hi? y
getting file hi of size 10 as hi (1.22069 kb/s) (average 1.2207 kb/s)
smb: \> quit
```

Once logged in, you can execute smbclient commands to manage files and change directories. The smbclient commands are listed in Table 38-5.

Table 38-5: smbclient Commands

Command	**Description**
? *[command]*	With no command argument, lists of all available commands are displayed. Use command argument to display information about a particular command.
! *[shell command]*	With no shell command argument, runs a local shell. If a shell command is provided, it executes that command.
cd *[directory name]*	Change directory on server. With no directory specified, the name of the current working directory is displayed.
del *mask*	Request the server delete all files matching mask from the current working directory on the server.
dir *mask*	A list of the files matching the mask in the current working directory on the server is retrieved from the server and displayed.
exit	Terminate the connection with the server and exit from the program.

Table 38-5: smbclient Commands

Command	**Description**
get *remote filename [local filename]*	Copy a file from the server to the local system. You can rename the local system copy. Transfer is binary.
help *[command]*	With no command argument, lists of all available commands are displayed. Use command argument to display information about a particular command. Same as **!**.
lcd *[directory name]*	Change directories on the local system. With no argument, the local directory name is displayed.
lowercase	Toggle lowercasing of filenames for the **get** and **mget** commands. When lowercasing is toggled ON, local filenames are converted to lowercase when using the **get** and **mget** commands. This is often useful when copying (say) MS-DOS files from a server because lowercase filenames are the norm on Unix systems.
ls *mask*	A list of the files matching the mask in the current working directory on the server is retrieved from the server and displayed. Same as **dir**.
mask *mask*	This command enables the user to set up a mask that is used during recursive operation of the **mget** and **mput** commands. The masks specified to the **mget** and **mput** commands act as filters for directories, rather than files when recursion is toggled ON. The value for mask defaults to blank (equivalent to *****) and remains so until the **mask** command is used to change it.
md *directory name*	Create a new directory on the server (user access privileges permitting) with the specified name. Same as **mkdir**.
mget *mask*	Copy all files matching mask from the server to the machine running the client. Note, mask is interpreted differently during recursive operation and nonrecursive operation—refer to the **recurse** and **mask** commands for more information. Transfers are binary.
mkdir *directory name*	Create a new directory on the server (user access privileges permitting) with the specified name.
mput *mask*	Copy all files matching mask in the current working directory on the local system to the current working directory on the server. Transfers in are binary.
print *filename*	Print the specified file from the local machine through a printable service on the server.
printmode *graphics or text*	Set the print mode for either binary data (graphics) or text. Subsequent print commands use the currently set print mode.
prompt	Toggle prompting for filenames during operation of the **mget** and **mput** commands.
put *local filename [remote filename]*	Copy a file on the local system to the server. You can rename the server copy. Transfers are binary.

Table 38-5: smbclient Commands

Command	Description
queue	Displays the print queue, showing the job ID, name, size, and current status.
quit	Terminate the connection with the server and exit from the program. Same as **exit** command.
rd *directory name*	Delete the specified directory (user access privileges permitting) from the server. Same as **rmdir** command.
recurse	Toggle directory recursion for the commands **mget** and **mput**. When toggled ON, the **mget** and **mput** commands will copy any subdirectories and files. Files can be selected by a mask specified with the **mget** and **mput** commands. The mask for directories is specified with the **mask** command. When toggled OFF, only files from the current working directory are copied.
rm *mask*	Delete all files matching mask from the current working directory on the server.
rmdir *directory name*	Delete the specified directory (user access privileges permitting) from the server.
tar *c*|*x* [IXbgNa]	Performs a **tar** operation. Behavior may be affected by the **tarmode** command.
blocksize *blocksize*	Specify block size. Must be followed by a valid (greater than zero) block size. Causes **tar** file to be written out in blocksize*TBLOCK (usually 512 byte) blocks.
tarmode *full*|*inc*|*reset*| noreset	Changes **tar**'s behavior regarding archive bits. In full mode, **tar** backs up everything, regardless of the archive bit setting (this is the default mode). In incremental mode, **tar** only backs up files with the archive bit set. In reset mode, **tar** resets the archive bit on all files it backs up (implies read/write share).
setmode *filename perm=[+|\-]rsha*	A version of the DOS **attrib** command to set file permissions. For example: **setmode myfile +r** would make **myfile** read-only.

Shell commands can be executed with the **!** operator. To transfer files, you can use the **mget** and **mput** commands, much as they are used in the FTP program. The **recurse** command enables you to turn on recursion to copy whole subdirectories at a time. You can use file-matching operators, referred to here as *masks,* to select a certain collection of files. The file-matching (mask) operators are *****, **[]**, and **?** (see Chapter 7). The default mask is *****, which matches everything. The following example uses **mget** to copy all files with a **.c** suffix, as in **myprog.c**:

```
smb> mget *.c
```

During transfers, you can have smbclient either prompt you for each individual file, or simply transfer all the selected ones. The **prompt** command toggles this file, prompting on and off.

To access a particular printer on a remote system, enter the printer name as an argument to the **smbclient** command, followed by any options. In the following example, smbclient accesses the **myepson** printer on the **turtle.mytrek.com** system, using the **dylan** login name:

```
smbclient //turtle.mytrek.com/myepson -U dylan
```

Once connected, an smb prompt is displayed and you can use smbclient commands such as **print** to print files and **printmode** to specify graphics or text. In the next example, the user prints a file called **myfile**, after having accessed the **myepson** printer on **turtle.mytrek.com**:

```
smb> print myfile
```

smbmount

With the **smbmount** command, a Linux or a Unix client can mount a shared directory onto its local system. The syntax for the **smbmount** command is similar to the **smbclient** command, with many corresponding options. The **smbmount** command takes as its arguments the Samba server and shared directory, followed by the local directory to where you want to mount the directory. The following example mounts the **myreports** directory onto the **/mnt/myreps** directory on the local system:

```
smbmount //turtle.mytrek.com/myreports /mnt/myreps –U dylan
```

To unmount the directory, use the **smbumount** command with the local directory name, as shown here:

```
smbumount /mnt/myreps
```

To mount the home directory of a particular user on the server, specify the **homes** service and the user's login name. The following example mounts the home directory of the user **larisa** to the **/home/chris/larisastuff** directory on the local system:

```
smbmount //turtle.mytrek.com/homes /home/chris/larisastuff –U larisa
```

You can also use **smbmount** to mount shared folders on Windows clients. Just specify the computer name of the Windows client along with its folder. If the folder name contains spaces, enclose it in single quotes. In the following example, the user mounts the windata folder on lizard as the **/mylinux** directory. For a folder with access to anyone, just press ENTER at the password prompt.

```
$ smbmount //lizard/windata  /mylinux
Password:
$ ls /mylinux
_hi_mynewdoc.doc_myreport.txt
```

To unmount the shared folder when you are finished with it, use the **smbumount** command.

```
smbumount /mylinux
```

Instead of using **smbmount** explicitly you can use the **mount** command with the file system type **smbfs**. **mount** will then run the **/sbin/mount.smbfs** command, which will invoke smbclient to mount the file system:

```
mount -t smbfs //lizard/windata  /mylinux
```

You could also specify a username and password as options, if user level access is required:

```
mount -t smbfs -o username=chris passwd=mypass //lizard/windata /mylinux
```

You can also use the smbfs type in an **/etc/fstab** entry to have a samba file system mounted automatically:

```
//lizard/windata /mylinux smbfs defaults 0 0
```

Windows Clients

To access Samba resources from a Windows system, you will need to make sure that your Windows system has enabled TCP/IP networking. This may already be the case if your Windows client is connected to the Microsoft network. If you need to connect a Windows system directly to a TCP/IP network that your Linux Samba server is running on, you should check that TCP/IP networking is enabled on that Windows system. This involves making sure that the Microsoft Network client and the TCP/IP protocol are installed, and that your network interface card (NIC adaptor) is configured to use TCP/IP. The procedures differ slightly on Windows 2000 and XP, and those for Windows 95, 98, and ME.

Once connected, your Samba shares and printers will appear in the Windows network window. Here you can access those shares and printers. If you are going to use a share frequently, you can assign it a disk label. Right-click on the share and select Map Network Drive, and select the drive label you want to use. On XP you can further select whether you want the drive automatically mounted when you log in.

Windows 95, 98, and ME

For Windows 95, 98, and ME, you click on a Network icon in the Control Panels window. Here you will see the network components loaded for your system. Check to make sure that Client for Microsoft Networks and TCP/IP protocol are installed. If not, click the Add button, select Client or Protocol, and then select the needed client or protocol. For example, to add the TCP/IP protocol, click Add, then click Protocol, and then select TCP/IP from the list of protocols. You can also allow the Windows system to share its own files and printers by selecting "File and Print Sharing".

Then check the computer's name and workgroup by selecting the Identification panel. Make sure your Samba entry for the workgroup matches it.

Then select the kind of access control for sharing you set up for Samba, either user level or share level. Click on the Access Control tab and select either Share-level or User-level access.

Then open the Network Neighborhood icon on your desktop. This will list your Samba server. If not, open Entire network and then Workgroups to find it. When you open your Samba server icon, the shared printers and directories set up on your Samba server should be displayed.

Windows 2000 and XP

On Windows 2000 open the Network and Dialup Connections window in the Control Panels, and in Windows XP open the Network Connections window also in Control Panels. If you do not already have a LAN connection, create one with the Make New Connection tool. Each NIC card will have its own Local Area Connections file (you probably have only one). Right-click on the one for the NIC adapter connected to the TCP/IP network that your Linux Samba server is on. Select Properties from the pop-up menu. The General tab will show the NIC card's name (adapter) and the components loaded and checked in support for this connection. Make sure the following are listed and checked:

- Client for Microsoft Networks
- File and Printer Sharing for Microsoft Networks
- Internet Protocol (TCP/IP)

Should any of these be missing, you can click on the Install button to install them. From a dialog box listing Client, Service, and Protocol, select the type of component you want to install. Then a list of components is displayed. For example, to install TCP/IP, click on the Install button, and then the Protocol entry in the Component Type window, and then the TCP/IP protocol listed in the Select Network Protocol window.

File and printer sharing enables access to the directories and printers on the Windows client. It is not essential for accessing shares and printers on the Samba server. However, if you want a user to be able to access, through Samba, a printer on a Windows client, then this feature has to be enabled on that Windows client.

Make sure that the IP address for this adapter is entered. This is the IP address for your Windows host. Click on the TCP/IP entry and then click on the Properties button. This displays a window for entering the IP address for this NIC adapter. Click on the Advanced button and then click on the WINS tab. Also, make sure that the Enable LMHOSTS Lookup and the Enable NetBIOS over TCP/IP entries are checked.

Next, check to make sure that the computer name and workgroup are the same as that used in your Samba **smb.conf** file. Click on System in Control Panels and click on the Network Identification tab. The computer name and workgroup for the Windows system will be displayed. If none are given, or if you want to change it, click on the Properties button to open a window where you can enter a computer name and workgroup.

To access your Samba server, click on the My Network Places icon, and then the Computers Near Me icon. On Windows XP, you can open the My Computer window and select the Computers Near Me entry on the left panel. Your Samba server will then show up. Double-click it to display the shares available on your Samba server. You can also click the Entire Network icon and choose to show entire contents. Click the Microsoft Windows Network icon and then your Workgroup icon. Your Samba server will be listed along with your Windows client.

If you specified share-level security in your **smb.conf** file (or on the Globals page of SWAT), the shares available to you will be displayed. Normally you would use share-level security if you are supporting resources like printers that you do not want to require users to log in for each time they use them.

If you specified the user-level security in the **smb.conf** file or the SWAT Globals page, then a dialog will first appear that will prompt you for a user name and password. You need to be logged in on Windows as the same user that you will be using to access the Samba server. If you want to access the Samba server as the aleina user, you need to be logged in as the aleina user on your Windows system.

To see what resources on your Windows client can be shared, double-click the icon for your client in the Workgroup or Computers Near Me window. Any resources such as printers or folders that you specified should be shared will be listed.

To use a printer connected to the Samba server, the Windows client first has to connect to it. Click the printer's icon in the Samba server's window and select Connect. The first time you do this, you will be prompted to configure the printer for use on your Windows system. A window will let you select the driver needed for that printer. Now, when you print in a application, you can choose the remote Linux printer from your list of available printers. For example, to use a printer on a Samba server named myhp, click the myhp icon in the Samba server window and select connect. You will be asked to specify the driver to use for this printer. If it is an HP printer, select its model. The appropriate Windows driver for that printer will be installed on the Windows client. Now when you print, you can select the myhp printer and have your document printed on that Linux printer. For example, if the Samba server's name is rabbit and the printer's name is myhp, you will see an entry like this among your list of printers.

```
\\rabbit\myhp
```

Mounting Linux Shares and Printers on Windows Clients

Your Samba shares and printers will appear in your Window network window. You can access them through here. You could also mount a share as a disk on your Windows system. The share will appear with a disk label, just as your other disks are listed. You can mount shares either with a **net** command entered on a command window or by right-clicking the share and selecting "Map Network Drive". As noted previously, selecting Map Network Drive will open a window where you can choose a disk label to use for the share. You can also specify if you want to have the share automatically mounted as that disk whenever you log in.

With the **net** command, you give a share a disk label with which you can reference it on your Windows system. The syntax for the command is as follows (notice that backslashes are used on Windows instead of the forward slashes used on Linux):

```
net use label: \\server\service
```

For example, to mount the **myprojects** directory on the Samba server **turtle.mytrek.com** as the h: disk, you would use the following command.

```
net use h: \\turtle.mytrek.com\myprojects
```

To print, you would specify the remote printer and the local port to use it as follows:

```
net use lpt1: \\turtle.trek.com\myhp
```

Then use the **print** command to print a file:

```
print filename
```

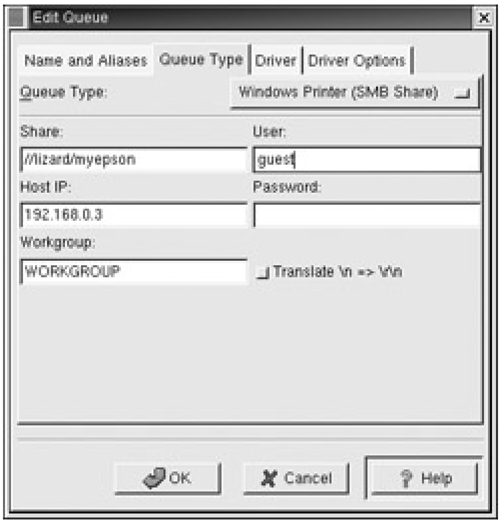
Sharing Windows Directories and Printers

To manage directory shares, open the Computer Management tool in the Administrative window in Control Panels. Click on Shared Folders and there you can see the Shares, Sessions, and Open folders. To add a new share, click on the Shares folder and then click the Action menu and select New File Share. The Sessions and Open folders' Action menus let you disconnect active sessions and folders.

To allow share-level open access by users on other clients or on the Samba server, be sure to enable the guest user on your Windows client. They are not enabled by default. Access the Users and Passwords tool in the Control Panels to a set up the guest user. Guest access is particularly important for providing access to a printer connected to a Windows client. The Linux system that wants to access a printer on Windows will configure the printer on its own system as a remote Samba printer. Users normally access the printer as the guest user. For the Linux system to access the Windows printer, that Windows system has to have a guest user.

To share a printer, locate the printer in the Printers window and right-click it, selecting the Sharing As option. This opens the Sharing panel where you can click the Shared As button and enter the name under which the printer will be known by other hosts. For example, on the Windows client named lizard, to have a printer called Epson Stylus Color shared as myepson, the Sharing panel for this printer would have the Shared As button selected and the name myepson entered. Then when the user double-clicks the lizard icon in the Computers Near Me window, the printer icon labeled myepson will appear.

For a Linux system to use this printer, it will have be first configured as a remote Windows printer on that Linux system. You can do this easily with printconf. You give the printer a name by which it is known on your Linux system. Then you will enter the Windows client computer name, the name of the printer as it is accessed on the Windows client, along with the user name for access (usually guest). Once configured, your printing commands can access it using just the printer name, as they would any other printer. For example, the myepson printer installed on the Windows client has to also be installed on the Linux system operating as the Samba server (see Figure 38-8). Using printconf, you can give the printer the same name, if you wish, and then in the Queue Type panel select Windows Printer (SMB Share). For Share, you enter **//lizard/myepson**, for User enter **guest**, and for Workgroup enter the Windows client's workgroup (usually WORKGROUP).



Queue Type: Windows Printer (SMB Share)

Share: //lizard/myepson

User: guest

Host IP: 192.168.0.3

Password:

Workgroup: WORKGROUP

□ Translate \n => \r\n

Figure 38-8: A Windows printer on Linux, printconf

Once installed, you can restart the lpd server. Then an **lpr** command can access the remote Windows printer directly. The next example prints the mydoc file on the Windows client's Epson printer.

```
lpr -P myepson mydoc
```

To share a directory, right-click on the directory and select Sharing from the pop-up menu (Sharing and Security on Windows XP). Click "Share this folder" and then enter the share name, the name by which the directory will be known by Samba. You can also specify whether you want to allow others to change files on the share. You can also specify a user limit (maximum allowed is the default). You can further click on the Permissions button to control access by users. Here, you can specify which users will have access, as well as the type of access. For example, you could allow only read access to the directory.

Chapter 39: Administering TCP/IP Networks

Overview

Linux systems are configured to connect into networks that use the TCP/IP protocols. These are the same protocols that the Internet uses, as do many local area networks (LANs). In Chapter 23, you were introduced to TCP/IP, a robust set of protocols designed to provide communications among systems with different operating systems and hardware. The TCP/IP protocols were developed in the 1970s as a special DARPA project to enhance communications between universities and research centers. These protocols were originally developed on Unix systems, with much of the research carried out at the University of California, Berkeley. Linux, as a version of Unix, benefits from much of this original focus on Unix. Currently, the TCP/IP protocol development is managed by the Internet Engineering Task Force (IETF), which, in turn, is supervised by the Internet Society (ISOC). The ISOC oversees several groups responsible for different areas of Internet development, such as the Internet Assigned Numbers Authority (IANA), which is responsible for Internet addressing (see Table 39-1). Over the years, TCP/IP protocol standards and documentation have been issued in the form of Requests for Comments (RFC) documents. Check the most recent ones for current developments at the IETF Web site at **www.ietf.org**.

Table 39-1: TCP/IP Protocol Development Groups

Group	**Title**	**Description**
ISOC	Internet Society	Professional membership organization of Internet experts that oversees boards and task forces dealing with network policy issues **www.isoc.org**
IESG	The Internet Engineering Steering Group	Responsible for technical management of IETF activities and the Internet standards process **www.ietf.org/iesg.html**
IANA	Internet Assigned Numbers Authority	Responsible for Internet Protocol (IP) addresses **www.iana.org**
IAB	Internet Architecture Board	Defines the overall architecture of the Internet, providing guidance and broad direction to the IETF **www.iab.org**
IETF	Internet Engineering Task Force	Protocol engineering and development arm of the Internet **www.ietf.org**

The TCP/IP protocol suite actually consists of different protocols, each designed for a specific task in a TCP/IP network. The three basic protocols are the Transmission Control Protocol (TCP), which handles receiving and sending out communications, the Internet Protocol (IP), which handles the actual transmissions, and the User Datagram Protocol (UPD), which also handles receiving and sending packets. The IP protocol, which is the base protocol that all others use, handles the actual transmissions, handling the packets of data with sender and receiver information in each. The TCP protocol is designed to work with cohesive messages or data. This protocol checks received packets and sorts them into their designated order, forming the original message. For data sent out, the TCP protocol breaks the data into separate packets, designating their order. The UDP protocol, meant to work on a much more raw level, also breaks down data into packets, but does not check their order. The TCP/IP protocol is designed to provide stable and reliable connections that ensure that all data is received and reorganized into its original order. UDP, on the other hand, is designed to simply send as much data as possible, with no guarantee that packets will all be received or placed in the proper order. UDP is often used for transmitting very large amounts of data of the type that can survive the loss of a few packets—for example, temporary images, video, and banners displayed on the Internet.

Other protocols provide various network and user services. For example, the Domain Name Service (DNS) provides address resolution. The File Transfer Protocol (FTP) provides file transmission, and Network File System (NFS) provides access to remote file systems. Table 39-2 lists the different protocols in the TCP/IP protocol suite. These protocols make use of either the TCP or UDP protocol to send and receive packets, which, in turn, uses the IP protocol for actually transmitting the packets.

Table 39-2: TCP/IP Protocol Suite

Transport	**Description**

Table 39-2: TCP/IP Protocol Suite

Transport	**Description**
TCP	Transmission Control Protocol; places systems in direct communication
UDP	User Datagram Protocol
IP	Internet Protocol; transmits data
ICMP	Internet Control Message Protocol; status messages for IP
Routing	Description
RIP	Routing Information Protocol; determines routing
OSPF	Open Shortest Path First; determines routing
Network Addresses	Description
ARP	Address Resolution Protocol; determines unique IP address of systems
DNS	Domain Name Service; translates hostnames into IP addresses
RARP	Reverse Address Resolution Protocol; determines addresses of systems
User Services	Description
FTP	File Transfer Protocol; transmits files from one system to another using TCP
TFTP	Trivial File Transfer Protocol; transfers files using UDP
TELNET	Remote login to another system on the network
SMTP	Simple Mail Transfer Protocol; transfers e-mail between systems
RPC	Remote Procedure Call; allows programs on remote systems to communicate
Gateway	Description
EGP	Exterior Gateway Protocol; provides routing for external networks
GGP	Gateway-to-Gateway Protocol; provides routing between Internet gateways
IGP	Interior Gateway Protocol; provides routing for internal networks
Network Services	Description
NFS	Network File System; allows mounting of file systems on remote machines
NIS	Network Information Service; maintains user accounts across a network
BOOTP	Boot Protocol; starts system using boot information on server for network
SNMP	Simple Network Management Protocol; provides status messages on TCP/IP configuration
DHCP	Dynamic Host Configuration Protocol; automatically provides network configuration information to host systems

In a TCP/IP network, messages are broken into small components, called *datagrams,* which are then transmitted through various interlocking routes and delivered to their destination computers. Once received, the datagrams are reassembled into the original message. Datagrams themselves can be broken down into smaller packets. The *packet* is the physical message unit actually transmitted among networks. Sending messages as small components has proved to be far more reliable and faster than sending them as one large, bulky transmission. With small components, if one is lost or damaged, only that component must be resent, whereas if any part of a large transmission is corrupted or lost, the entire message has to be resent.

The configuration of a TCP/IP network on your Linux system is implemented using a set of network configuration files (Table 39-5 provides a complete listing). Many of these can be managed using administrative programs, such as Linuxconf or netcfg**,** on your root user desktop (see Chapter 29). You can also use the more specialized programs, such as netstat, ifconfig, and route. Some configuration files are easy to modify yourself using a text editor.

TCP/IP networks are configured and managed with a set of utilities: ifconfig, route, and netstat. The ifconfig utility operates from your root user desktop and enables you to configure your network interfaces fully, adding new ones and modifying others. The ifconfig and route utilities are lower-level programs that require more specific knowledge of your network to use effectively. The netstat utility provides you with information about the status of your network connections.

IPv4 and IPv6

Traditionally, a TCP/IP address is organized into four segments, consisting of numbers separated by periods. This is called the *IP address.* The IP address actually represents a 32-bit integer whose binary values identify the network and host. This form of IP addressing adheres to Internet Protocol, version 4, also known as IPv4. IPv4, the kind of IP addressing described here, is still in wide use.

Currently, a new version of the IP protocol called Internet Protocol, version 6 (IPv6) is gradually replacing the older IPv4 version. IPv6 expands the number of possible IP addresses using a 128-bit address. It is fully compatible with systems still using IPv4. IPv6 addresses are represented differently, using a set of eight 16-bit segments, each separated by a colon. Each segment is represented by a hexadecimal number. A sample address would be

```
FEDC:0:0:200C:800:BA98:7654:3210
```

IPv6 features simplified headers that allow for faster processing. It also provides support for encryption and authentication. Its most significant advantage is extending the address space to cover 2 to the power of 128 possible hosts (billions of billions of billions—a lot). This extends far beyond the 4.2 billion supported by IPv4.

TCP/IP Network Addresses

As noted previously, the traditional IPv4 TCP/IP address is organized into four segments, consisting of numbers separated by periods. This kind of address is still in wide use and is what people commonly refer to as an *IP address.* Part of an IP address is used for the network address, and the other part is used to identify a particular interface on a host in that network.

You should realize that IP addresses are assigned to interfaces—such as Ethernet cards or modems—and not to the host computer. Usually a computer has only one interface and is accessed using only that interface's IP address. In that regard, an IP address can be thought of as identifying a particular host system on a network, and so the IP address is usually referred to as the *host address.*

In fact, though, a host system could have several interfaces, each with its own IP address. This is the case for computers that operate as gateways and firewalls from the local network to the Internet. One interface usually connects to the LAN and another to the Internet, as by two Ethernet cards. Each interface (such as an Ethernet card) has its own IP address. For example, when you use Linuxconf to specify an IP address for an Ethernet card on your system, the panel for entering your IP address is labeled as Adaptor 1, and three other panels are there for other Ethernet cards that have their own IP addresses. Currently, the Linux kernel can support up to four network adapters. If you use a modem to connect to an ISP, you would set up a PPP interface that would also have its own IP address (usually dynamically assigned by the ISP). Remembering this distinction is important if you plan to use Linux to set up a local or home network, using Linux as your gateway machine to the Internet (see the section "IP Masquerading" later in Chapter 40).

Network Addresses

The IP address is divided into two parts: one part identifies the network, and the other part identifies a particular host. The network address identifies the network of which a particular interface on a host is a part. Two methods exist for implementing the network and host parts of an IP address: the original class-based IP addressing and the current Classless Interdomain Routing (CIDR) addressing. Class-based IP addressing designates officially predetermined parts of the address for the network and host addresses, whereas CIDR addressing allows the parts to be determined dynamically using a netmask.

Class-Based IP Addressing

Originally, IP addresses were organized according to classes. On the Internet, networks are organized into three classes depending on their size—classes A, B, and C. A class A network uses only the first segment for the network address and the remaining three for the host, allowing a great many computers to be connected to the same network. Most IP addresses reference smaller, class C, networks. For a class C network, the first three segments are used to identify the network, and only the last segment identifies the host. Altogether, this forms a unique address with which to identify any network interface on computers in a TCP/IP network. For example, in the IP address 192.168.1.72, the network part is 192.168.1 and the interface/host part is 72. The interface/host is a part of a network whose own address is 192.168.1.0.

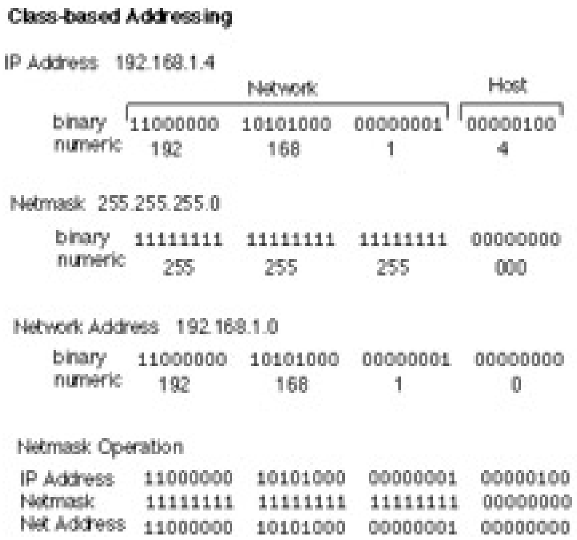
In a class C network, the first three numbers identify the network part of the IP address. This part is divided into three network numbers, each identifying a subnet. Networks on the Internet are organized into subnets, beginning with the largest and narrowing to small subnetworks. The last number is used to identify a particular computer, referred to as a *host.* You can think of the Internet as a series of networks with subnetworks; these subnetworks have their own subnetworks. The rightmost number identifies the host computer, and the number preceding it identifies the subnetwork of which the computer is a part. The number to the left of that identifies the network the subnetwork is part of, and so on. The Internet address

192.168.187.4 references the fourth computer connected to the network identified by the number 187. Network 187 is a subnet to a larger network identified as 168. This larger network is itself a subnet of the network identified as 192. Here's how it breaks down:

192.168.187.4	IP address
192.168.187	Network identification
4	Host identification

Netmask

Systems derive the network address from the host address using the netmask. You can think of an IP address as a series of 32 binary bits, some of which are used for the network and the remainder for the host. The *netmask* has the network set of bits set to 1s, with the host bits set to 0s (see Figure 39-1). In a standard class-based IP address, all the numbers in the network part of your host address are set to 255, and the host part is set to 0. This has the effect of setting all the binary bits making up the network address to 1s. This, then, is your netmask. So, the netmask for the host address 192.168.1.72 is 255.255.255.0. The network part, 192.168.1, has been set to 255.255.255, and the host part, 72, has been set to 0. Systems can then use your netmask to derive your network address from your host address. They can determine what part of your host address makes up your network address and what those numbers are.



```
Class-based Addressing

IP Address  192.168.1.4
                     Network                      Host
     binary   11000000  10101000  00000001    00000100
     numeric    192       168        1           4

Netmask  255.255.255.0
     binary   11111111  11111111  11111111  00000000
     numeric    255       255       255       000

Network Address  192.168.1.0
     binary   11000000  10101000  00000001  00000000
     numeric    192       168        1          0

Netmask Operation
 IP Address   11000000  10101000  00000001  00000100
 Netmask      11111111  11111111  11111111  00000000
 Net Address  11000000  10101000  00000001  00000000
```

Figure 39-1: Class-based netmask operations

For those familiar with computer programming, a bitwise AND operation on the netmask and the host address results in zeroing the host part, leaving you with the network part of the host address. You can think of the address as being implemented as a four-byte integer with each byte corresponding to a segment of the address. In a class C address, the three network segments correspond to the first three bytes and the host segment corresponds to the fourth byte. A netmask is designed to mask out the host part of the address, leaving the network segments alone. In the netmask for a standard class C network, the first three bytes are all 1s and the last byte consists of 0s. The 0s in the last byte mask out the host part of the address, and the 1s in the first three bytes leave the network part of the address alone. Figure 39-1 shows the bitwise operation of the netmask on the address 192.168.1.4. This is a class C address to the mask, which consists of twenty-four 1s making up the first three bytes and eight

0s making up the last byte. When it is applied to the address 192.168.1.4, the network address remains (192.168.1) and the host address is masked out (4), giving you 192.168.1.0 as the network address.

The netmask as used in Classless Interdomain Routing (CIDR) is much more flexible. Instead of having the size of the network address and its mask determined by the network class, it is determined by a number attached to the end of the IP address. This number simply specifies the size of the network address, how many bits in the address it takes up. For example, in an IP address whose network part takes up the first three bytes (segments), the number of bits used for that network part is 24—eight bits to a byte (segment). Instead of using a netmask to determine the network address, the number for the network size is attached to the end of the address with a slash, as shown here:

```
192.168.1.72/24
```

CIDR gives you the advantage of specifying networks that are any size bits, instead of only three possible segments. You could have a network whose addresses take up 14 bits, 22 bits, or even 25 bits. The host address can use whatever bits are left over. An IP address with 21 bits for the network can cover host addresses using the remaining 11 bits, 0 to 2,047.

Classless Interdomain Routing (CIDR)

Currently, the class-based organization of IP addresses is being replaced by the CIDR format. CIDR was designed for midsized networks, those between a class C and classes with numbers of hosts greater than 256 and smaller than 65,534. A class C network–based IP address using only one segment for hosts uses only one segment, an 8-bit integer, with a maximum value of 256. A class B network–based IP address uses two segments, which make up a 16-bit integer whose maximum value is 65,534. You can think of an address as a 32-bit integer taking up four bytes, where each byte is eight bits. Each segment conforms to one of the four bytes. A class C network uses three segments, or 24 bits, to make up its network address. A class B network, in turn, uses two segments, or 16 bits, for its address. With this scheme, allowable host and network addresses are changed an entire byte at a time, segment to segment. With CIDR addressing, you can define host and network addresses by bits, instead of whole segments. For example, you can use CIDR addressing to expand the host segment from eight bits to nine, rather than having to jump it to a class B 16 bits (two segments).

CIDR addressing notation achieves this by incorporating netmask information in the IP address (the netmask is applied to an IP address to determine the network part of the address). In the CIDR notation, the number of bits making up the network address is placed after the IP address, following a slash. For example, the CIDR form of the class C 192.168.187.4 IP address is

```
192.168.187.4/24
```

Figure 39-2 shows an example of a CIDR address and its network mask. The IP address is 192.168.1.6 with a network mask of 22 bits, 192.168.1.6/22. The network address takes up the first 22 bits of the IP address, and the remaining 10 bits are used for the host address. The host address is taking up the equivalent of a class-based IP address's fourth segment (8 bits) and two bits from the third segment.

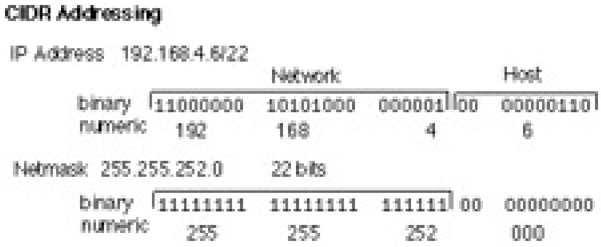

CIDR Addressing
IP Address 192.168.4.6/22
Network
Host
binary 11000000 10101000 000001 00 00000110
numeric 192 168 4 6
Netmask 255.255.252.0
22 bits
binary 11111111 11111111 111111 00 00000000
numeric 255 255 252 000

Figure 39-2: CIDR addressing

Table 39-3 lists the different CIDR network masks available along with the maximum number of hosts. Both the short form and the full forms of the netmask are listed.

Table 39-3: CIDR Network Masks

Short Form	Full Form	Maximum Number of Hosts
/8	/255.0.0.0	16,777,215 (A class)
/16	/255.255.0.0	65,535 (B class)
/17	/255.255.128.0	32,767
/18	/255.255.192.0	16,383
/19	/255.255.224.0	8,191
/20	/255.255.240.0	4,095
/21	/255.255.248.0	2,047
/22	/255.255.252.0	1,023
/23	/255.255.254.0	511
/24	/255.255.255.0	255 (C class)
/25	/255.255.255.128	127
/26	/255.255.255.192	63
/27	/255.255.255.224	31
/28	/255.255.255.240	15
/29	/255.255.255.248	7
/30	/255.255.255.252	3

The network address for any standard class C IP address takes up the first three segments, 24 bits. If you want to create a network with a maximum of 512 hosts, you can give them IP addresses where the network address is 23 bits and the host address takes up 9 bits (0–511). The IP address notation remains the same, however, using the four 8-bit segments. This means a given segment's number could be used for both a network address and a host address. Segments are no longer wholly part of either the host address or the network address. Assigning a 23-bit network address and a 9-bit host address means that the number in the third segment is part of both the network address and the host address, the first seven bits for the network and the last bit for the host. In this following example, the third number, 145, is used as the end of the network address and as the beginning of the host address:

```
192.168.145.67/23
```

This situation complicates CIDR addressing, and in some cases the only way to represent the address is to specify two or more network addresses. Check RFC 1520 at www.ietf.org for more details.

Note A simple way to calculate the number of hosts a network can address is to take the number of bits in its host segment as a power of 2, then subtract 2—that is, 2 to the number of host bits, minus 2. For example, an 8-bit host segment would be 2 to the power of 8, which equals 256. Subtract 2, one for the broadcast address, 255, and one for the zero value, 000, to leave you with 254 possible hosts.

CIDR also allows a network administrator to take what is officially the host part of an IP address and break it up into subnetworks with fewer hosts. This is referred to as *subnetting*. A given network will have its official IP network address recognized on the Internet or by a larger network. The network administrator for that network could, in turn, create several smaller networks within it using CIDR network masking. A classic example is to take a standard class C network with 254 hosts and break it up into two smaller networks, each with 64 hosts. You do this by using a CIDR netmask to take a bit from the host part of the IP address and use it for the subnetworks. Numbers within the range of the original 254 addresses whose first bit would be set to 1 would represent one subnet, and the others, whose first bit would be set to 0, would constitute the remaining network. In the network whose network address is 192.168.187.0, where the last segment is used for the hostnames, that last host segment could be further split into two subnets, each with its own hosts. For two subnets, you would use the first bit in the last 8-bit segment for the network. The remaining seven bits could then be used for host addresses, giving you a range of 127 hosts per network. The subnet whose bit is set to 0 would have a range of 1 to 127, with a CIDR netmask of 25. The 8-bit segment for the first host would be 00000001. So, the host with the address of 1 in that network would have this IP address:

```
192.168.187.1/25
```

For the subnet where the first bit is 1, the first host would have an address of 129, with the CIDR netmask of 25, as shown here. The 8-bit sequence for the first host would be 10000001.

```
192.168.187.129/25
```

Each subnet would have a set of 126 addresses, the first from 1 to 126, and the second from 129 to 254; 127 is the broadcast address for the first subnet, and 128 is the network address for the second subnet. The possible subnets and their masks that you could use are shown here.

Subnetwork	**CIDR Address**	**Binary Mask**
First subnet network address	.0/25	00000000
Second subnet network address	.128/25	10000000
First subnet broadcast address	.127/25	01111111
Second subnet broadcast address	.255/25	11111111
First address in first subnet	.1/25	00000001
First address in second subnet	.129/25	10000001
Last address in first subnet	.126/25	01111110

Subnetwork	CIDR Address	Binary Mask
Last address in second subnet	.254/25	11111110

Obtaining an IP Address

IP addresses are officially allocated by IANA, which manages all aspects of Internet addressing (**www.iana.org**). IANA oversees Internet Registries (IRs), which, in turn, maintain Internet addresses on regional and local levels. The Internet Registry for the Americas is the American Registry for Internet Numbers (ARIN), whose Web site is at **www.arin.net.** These addresses are provided to users by Internet service providers (ISPs). You can obtain your own Internet address from an ISP, or if you are on a network already connected to the Internet, your network administrator can assign you one. If you are using an ISP, the ISP may temporarily assign one from a pool it has on hand with each use.

Certain numbers are reserved. The numbers 127, 0, or 255 cannot be part of an official IP address. The number 127 is used to designate the network address for the loopback interface on your system. The loopback interface enables users on your system to communicate with each other within the system without having to route through a network connection. Its network address would be 127.0.0.0 and its IP address is 127.0.0.1. For class-based IP addressing, the number 255 is a special broadcast identifier you can use to broadcast messages to all sites on a network. Using 255 for any part of the IP address references all nodes connected at that level. For example, 192.168.255.255 broadcasts a message to all computers on network 192.168, all its subnetworks, and their hosts. The address 192.168.187.255 broadcasts to every computer on the local network. If you use 0 for the network part of the address, the host number references a computer within your local network. For example, 0.0.0.6 references the sixth computer in your local network. If you want to broadcast to all computers on your local network, you can use the number 0.0.0.255. For CIDR IP addressing, the broadcast address may appear much like a normal IP address. As indicated in the previous section, CIDR addressing allows the use of any number of bits to make up the IP address for either the network or the host part. For a broadcast address, the host part must have all its bits set to 1 (see Figure 39-1).

A special set of numbers is reserved for use on non-Internet LANs (RFC 1918). These are numbers that begin with the special network number 192.168 (for class C networks), as used in these examples. If you are setting up a LAN, such as a small business or a home network, you are free to use these numbers for your local machines. You can set up an intranet using network cards, such as Ethernet cards and Ethernet hubs, and then configure your machines with IP addresses starting from 192.168.1.1. The host segment can go up to 256. If you have three machines on your home network, you could give them the addresses 192.168.1.1, 192.168.1.2, and 192.168.1.3. You can implement Internet services, such as FTP, Web, and mail services, on your local machines and use any of the Internet tools to make use of those services. They all use the same TCP/IP protocols used on the Internet. For example, with FTP tools, you can transfer files among the machines on your network; with mail tools, you can send messages from one machine to another; and with a Web browser, you can access local Web sites that may be installed on a machine running its own Web servers. If you want to have one of your machines connected to the Internet or some other network, you can set it up to be a gateway machine. By convention, the gateway machine is usually given the address 192.168.1.1. With a method called *IP masquerading,* you can have any of the non-Internet machines use a gateway to connect to the Internet.

Numbers are also reserved for class A and class B non-Internet local networks. Table 39-4 lists these addresses. The possible addresses available span from 0 to 255 in the host segment of the address. For example, class B network addresses range from 172.16.0.0 to 172.31.255.255, giving you a total of 32,356 possible hosts. The class C network ranges from 192.168.0.0 to 192.168.255.255, giving you 256 possible subnetworks, each with 256 possible hosts. The network address 127.0.0.0 is reserved for a system's loopback interface, which allows it to communicate with itself, enabling users on the same system to send messages to each other.

Table 39-4: Non-Internet Local Network IP Addresses

Private Network Address	Network Classes
10.0.0.0	Class A network
172.16.0.0 to 172.31.255.255	Class B network
192.168.0.0	Class C network
127.0.0.0	Loopback network (for system self-communication)

Broadcast Addresses

The broadcast address allows a system to send the same message to all systems on your network at once. With class-based IP addressing, you can easily determine the broadcast address using your host address: the broadcast address has the host part of your address set to 255. The network part remains untouched. So, the broadcast address for the host address 192.168.1.72 is 192.168.1.255 (you combine the network part of the address with 255 in the host part). For CIDR IP addressing, you need to know the number of bits in the netmask. The remaining bits are set to 1 (see Figure 39-3). For example, an IP address of 192.168.4.6/22 has a broadcast address of 192.168.7.255/22. In this case, the first 22 bits are the network address and the last 10 bits are the host part set to the broadcast value (all 1s).

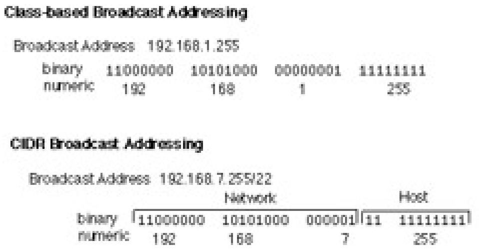

Class-based Broadcast Addressing
Broadcast Address 192.168.1.255
binary 11000000 10101000 00000001 11111111
numeric 192 168 1 255
CIDR Broadcast Addressing
Broadcast Address 192.168.7.255/22
Network
Host
binary 11000000 10101000 000001 11 11111111
numeric 192 168 7 255

Figure 39-3: Class-based and CIDR broadcast addressing

In fact, you can think of a class C broadcast address as merely a CIDR address using 24 bits (the first three segments) for the network address, and the last eight bits (the fourth segment) as the broadcast address. The value 255 expressed in binary terms is simply eight bits that are all 1s. 255 is the same as 11111111.

IP Address	Broadcast Address	IP Broadcast Number	Binary Equivalent
192.168.1.72	192.168.1.255	255	11111111
192.168.4.6/22	192.168.7.255/22	7.255 (last 2 bits in 7)	1111111111

Gateway Addresses

Some networks have a computer designated as the gateway to other networks. Every connection to and from a network to other networks passes through this gateway computer. Most local networks use gateways to establish a connection to the Internet. If you are on this type of network, you must provide the gateway address. If your network does not have a connection to the Internet, or a larger network, you may not need a gateway address. The gateway address is the address of the host system providing the gateway service to the network. On many networks, this host is given a host ID of 1: the gateway address for a network with the address 192.168.1 would be 192.168.1.1, but this is only a convention. To be sure of your gateway address, ask your network administrator.

Name Server Addresses

Many networks, including the Internet, have computers that provide a Domain Name Service (DNS) that translates the domain names of networks and hosts into IP addresses. These are known as the network's *domain name servers*. The DNS makes your computer identifiable on a network, using only your domain name rather than your IP address. You can also use the domain names of other systems to reference them, so you needn't know their IP addresses. You must know the IP addresses of any domain name servers for your network, however. You can obtain the addresses from your system administrator (often more than one exists). Even if you are using an ISP, you must know the address of the domain name servers your ISP operates for the Internet.

TCP/IP Configuration Files

A set of configuration files in the **/etc** directory, shown in Table 39-5, is used to set up and manage your TCP/IP network. These configuration files specify such network information as host and domain names, IP addresses, and interface options. The IP addresses and domain names of other Internet hosts you want to access are entered in these files. If you configured your network during installation, you can already find that information in these files. The netcfg, Linuxconf, and the netconfig configuration tools described in the next section provide easy interfaces for entering the configuration data for these files.

Identifying Hostnames: /etc/hosts

Without the unique IP address the TCP/IP network uses to identify computers, a particular computer cannot be located. Because IP addresses are difficult to use or remember, domain names are used instead. For each IP address, a domain name exists. When you use a domain name to reference a computer on the network, your system translates it into its associated IP address. This address can then be used by your network to locate that computer.

Originally, every computer on the network was responsible for maintaining a list of the hostnames and their IP addresses. This list is still kept in the **/etc/hosts** file. When you use a domain name, your system looks up its IP address in the **hosts** file. The system administrator is responsible for maintaining this list. Because of the explosive growth of the Internet and the development of more and more large networks, the responsibility for associating domain names and IP addresses has been taken over by domain name servers. The **hosts** file is still used to hold the domain names and IP addresses of frequently accessed hosts, however. Your system normally checks your **hosts** file for the IP address of a domain name before taking the added step of accessing a name server.

The format of a domain name entry in the **hosts** file is the IP address followed by the domain name, separated by a space. You can then add aliases for the hostname. After the entry, on the same line, you can enter a comment. A comment is always preceded by a # symbol. You can already find an entry in your **hosts** file for localhost with the IP address 127.0.0.1. Localhost is a special identification used by your computer to enable users on your system to communicate locally with each other. The IP address 127.0.0.1 is a special reserved address used by every computer for this purpose. It identifies what is technically referred to as a loopback device. A sample **/etc/hosts** file is shown here.

/etc/hosts

```
127.0.0.1         turtle.mytrek.com localhost
192.168.0.1       turtle.mytrek.com
192.168.0.2       rabbit.mytrek.com
192.168.34.56     pango1.mytrain.com
202.211.234.1     rose.berkeley.edu
```

Network Name: /etc/networks

The **/etc/networks** file holds the domain names and IP addresses of networks you are connected to, not the domain names of particular computers. Networks have shortened IP addresses. Depending on the type of network, they use one, two, or three numbers for their IP addresses. You also have your localhost network IP address 127.0.0.0. This is the network address used for the loopback device.

The IP addresses are entered, followed by the network domain names. Recall that an IP address consists of a network part and a host part. The network part is the network address you find in the **networks** file. You always have an entry in this file for the network portion of your computer's IP address. This is the network address of the network to which your computer is connected. A sample **/etc/networks** file is shown here with an entry for the **mytrek.com** network.

/etc/networks

```
loopback 127.0.0.0
mytrek.com 192.168.1.0
```

/etc/HOSTNAME

The **/etc/HOSTNAME** file holds your system's hostname. To change your hostname, you change this entry. The netcfg program enables you to change your hostname and places the new name in **/etc/HOSTNAME**. Instead of displaying this file to find your hostname, you can use the **hostname** command:

```
$ hostname
```

```
turtle.mytrek.com
```

/etc/services

The **/etc/services** file lists network services available on your system, such as FTP and telnet, and associates each with a particular port. Here, you can find out what port your Web server is checking or what port is used for your FTP server. You can give a service an alias, which you specify after the port number. You can then reference the service using the alias.

/etc/protocols

The **/etc/protocols** file lists the TCP/IP protocols currently supported by your system.

/etc/sysconfig/network

The **/etc/sysconfig/network** file contains system definitions for your network configuration. These include definitions for your domain name, gateway, and hostname, as shown here:

```
NETWORKING=yes
HOSTNAME=turtle.mytrek.com
GATEWAY=192.168.0.1
```

Table 39-5: TCP/IP Configuration Addresses and Files

Address	**Description**
Host address	IP address of your system; it has a network part to identify the network you are on and a host part to identify your own system
Network address	IP address of your network
Broadcast address	IP address for sending messages to all hosts on your network at once
Gateway address	IP address of your gateway system, if you have one (usually the network part of your host IP address with the host part set to 1)
Domain name server addresses	IP addresses of domain name servers your network uses
Netmask	Used to determine the network and host parts of your IP address
Files	**Description**
/etc/hosts	Associates hostnames with IP addresses
/etc/networks	Associates domain names with network addresses
/etc/host.conf	Lists resolver options
/etc/nsswitch.conf	Name Service Switch configuration file
/etc/hosts	Lists domain names for remote hosts with their IP addresses
/etc/resolv.conf	Lists domain name server names, IP addresses (nameserver), and domain names where remote hosts may be located (search)

Table 39-5: TCP/IP Configuration Addresses and Files

Address	**Description**
/etc/protocols	Lists protocols available on your system
/etc/services	Lists available network services, such as FTP and telnet, and the ports they use
/etc/HOSTNAME	Holds the name of your system
/etc/sysconfig/network	Network configuration information

Domain Name Service (DNS)

Each computer connected to a TCP/IP network, such as the Internet, is identified by its own IP address. IP addresses are difficult to remember, so a domain name version of each IP address is also used to identify a host. As described in Chapter 10, a domain name consists of two parts, the hostname and the domain. The hostname is the computer's specific name, and the domain identifies the network of which the computer is a part. The domains used for the United States usually have extensions that identify the type of host. For example, **.edu** is used for educational institutions and **.com** is used for businesses. International domains usually have extensions that indicate the country they are located in, such as **.de** for Germany or **.au** for Australia. The combination of a hostname, domain, and extension forms a unique name by which a computer can be referenced. The domain can, in turn, be split into further subdomains.

As you know, a computer on a network can still only be identified by its IP address, even if it has a hostname. You can use a hostname to reference a computer on a network, but this involves using the hostname to look up the corresponding IP address in a database. The network then uses the IP address, not the hostname, to access the computer. Before the advent of large TCP/IP networks, such as the Internet, it was feasible for each computer on a network to maintain a file with a list of all the hostnames and IP addresses of the computers connected on its network. Whenever a hostname was used, it was looked up in this file and the corresponding IP address was located. You can still do this on your own system for remote systems you access frequently.

As networks became larger, it became impractical—and, in the case of the Internet, impossible—for each computer to maintain its own list of all the domain names and IP addresses. To provide the service of translating domain addresses to IP addresses, databases of domain names were developed and placed on their own servers. To find the IP address of a domain name, you send a query to a name server, which then looks up the IP address for you and sends it back. In a large network, several name servers can cover different parts of the network. If a name server cannot find a particular IP address, it sends the query on to another name server that is more likely to have it.

If you are administering a network and you need to set up a name server for it, you can configure a Linux system to operate as a name server. To do so, you must start up a name server daemon and then wait for domain name queries. A name server makes use of several configuration files that enable it to answer requests. The name server software used on Linux systems is the Berkeley Internet Name Domain (BIND) server distributed by the Internet Software Consortium (**www.isc.org**). Chapter 18 describes the process of setting up a domain name server in detail.

Name servers are queried by resolvers. These are programs specially designed to obtain addresses from name servers. To use domain names on your system, you must configure your own resolver. Your local resolver is configured with your **/etc/host.conf** and **/etc/resolv.conf** files. You can use **/etc/nsswitch** in place of **/etc/host.conf**.

host.conf

Your **host.conf** file lists resolver options (shown in the following table). Each option can have several fields, separated by spaces or tabs. You can use a # at the beginning of a line to enter a comment. The options tell the resolver what services to use. The order of the list is important. The resolver begins with the first option listed and moves on to the next ones in turn. You can find the **host.conf** file in your **/etc** directory, along with other configuration files.

Option	**Description**
order	Specifies sequence of name resolution methods: **hosts** Checks for name in the local /etc/host file **bind** Queries a DNS name server for address **nis** Uses Network Information Service protocol to obtain address
alert	Checks addresses of remote sites attempting to access your system; you turn it on or off with the on and off options
nospoof	Confirms addresses of remote sites attempting to access your system
trim	Checks your local host's file; removes the domain name and checks only for the hostname; enables you to use only a hostname in your host file for an IP address
multi	Checks your local hosts file; allows a host to have several IP addresses; you turn it on or off with the on and off options

In the next example of a **host.conf** file, the **order** option instructs your resolver first to look up names in your local **/etc/hosts** file, and then, if that fails, to query domain name servers. The system does not have multiple addresses.

/etc/host.conf

```
# host.conf file
# Lookup names in host file and then check DNS
order bind host
# There are no multiple addresses
multi off
```

/etc/nsswitch.conf: Name Service Switch

Different functions in the standard C Library must be configured to operate on your Linux system. Previously, database-like services, such as password support and name services like NIS or DNS, directly accessed these functions, using a fixed search order. For GNU C Library 2*.x,* used on current versions of Linux, this configuration is carried out by a scheme called the Name Service Switch (NSS), which is based on the method of the same name used by Sun Microsystems Solaris 2 OS. The database sources and their lookup order are listed in the **/etc/nsswitch.conf** file.

The **/etc/nsswitch.conf** file holds entries for the different configuration files that can be controlled by NSS. The system configuration files that NSS supports are listed in Table 39-6. An entry consists of two fields: the service and the configuration specification. The service consists of the configuration file followed by a colon. The second field is the configuration specification for that file, which holds instructions on how the lookup procedure will work. The configuration specification can contain service specifications and action items. Service specifications are the services to search. Currently, valid service specifications are nis, nis-plus, files, db, dns, and compat (see Table 39-7). Not all are valid for each configuration file. For example, the dns service is only valid for the **hosts** file, whereas nis is valid for all files. An action item specifies the action to take for a specific service. An action item is placed within brackets after a service. A configuration specification can list several services, each with its own action item. In the following example, the entry for the network file has a configuration specification that says to check the NIS service and, if not found, to check the **/etc/networks** file:

```
networks: nis [NOTFOUND=return] files
```

Table 39-6: NSS Supported Files

File	**Description**
aliases	Mail aliases, used by Sendmail
ethers	Ethernet numbers
group	Groups of users
hosts	Hostnames and numbers
netgroup	Network-wide list of hosts and users, used for access rules; C libraries before glibc 2.1 only support netgroups over NIS
network	Network names and numbers
passwd	User passwords
protocols	Network protocols
publickey	Public and secret keys for SecureRPC used by NFS and NIS+
rpc	Remote procedure call names and numbers
services	Network services
shadow	Shadow user passwords

Table 39-7: NSS Configuration Services

Service	**Description**
files	Check corresponding **/etc** file for the configuration (for example, **/etc/hosts** for hosts); this service is valid for all files
db	Check corresponding **/var/db** databases for the configuration; valid for all files except **netgroup**
compat	Valid only for **passwd**, **group**, and **shadow** files
dns	Check the DNS service; valid only for **hosts** file
nis	Check the NIS service; valid for all files

Table 39-7: NSS Configuration Services	
Service	**Description**
nisplus	NIS version 3
hesiod	Use Hesiod for lookup

An action item consists of a status and an action. The status holds a possible result of a service lookup, and the action is the action to take if the status is true. Currently, the possible status values are SUCCESS, NOTFOUND, UNAVAIL, and TRYAGAIN (service temporarily unavailable). The possible actions are **return** and **continue**: **return** stops the lookup process for the configuration file, whereas **continue** continues on to the next listed service. In the previous example, if the record is not found in NIS, the lookup process ends.

Shown here is a copy of the current Red Hat **/etc/nsswitch.conf** file. Comments and commented-out entries begin with a # sign.

/etc/nsswitch.conf

```
#
# /etc/nsswitch.conf
#
# An example Name Service Switch config file. This file should be
# sorted with the most-used services at the beginning.
#
# The entry '[NOTFOUND=return]' means that the search for an
# entry should stop if the search in the previous entry turned
# up nothing. Note that if the search failed due to some other reason
# (like no NIS server responding) then the search continues with the
# next entry.
#
# Legal entries are:
#
# nisplus or nis+   Use NIS+ (NIS version 3)
# nis or yp         Use NIS (NIS version 2), also called YP
# dns               Use DNS (Domain Name Service)
# files             Use the local files
# db                Use the local database (.db) files
# compat            Use NIS on compat mode
# hesiod            Use Hesiod for user lookups
# [NOTFOUND=return] Stop searching if not found so far
#

# To use db, put the "db" in front of "files" for entries you want to
# be looked up first in the databases
#
# Example:
#passwd: db files nisplus
#shadow: db files nisplus
#group: db files nisplus

passwd: files nisplus
shadow: files nisplus
group: files nisplus

hosts: files nisplus nis dns

# Example - obey only what nisplus tells us...
```

```
#services: nisplus [NOTFOUND=return] files
#networks: nisplus [NOTFOUND=return] files
#protocols: nisplus [NOTFOUND=return] files
#rpc: nisplus [NOTFOUND=return] files
#ethers: nisplus [NOTFOUND=return] files
#netmasks: nisplus [NOTFOUND=return] files

bootparams: nisplus [NOTFOUND=return] files

ethers: files
netmasks: files
networks: files
protocols: files nisplus
rpc: files
services: files nisplus

netgroup: files nisplus

publickey: nisplus

automount: files nisplus
aliases: files nisplus
```

Network Interfaces and Routes: ifconfig and route

Your connection to a network is made by your system through a particular hardware interface, such as an Ethernet card or a modem. Data passing through this interface is then routed to your network. The **ifconfig** command configures your network interfaces, and the **route** command sets up network connections accordingly. If you configure an interface with a network configuration tool, such as netcfg, Linuxconf, or YaST, you needn't use ifconfig or route. If you are using another Linux system, the netconfig utility also performs the same configuration as netcfg. However, you can directly configure interfaces using ifconfig and route, if you want. Every time you start your system, the network interfaces and their routes must be established. This is done automatically for you when you boot up by **ifconfig** and **route** commands executed for each interface by the **/etc/rc.d/init.d/network** initialization file, which is executed whenever you start your system. If you are manually adding your own interfaces, you must set up the network script to perform the ifconfig and route operations for your new interfaces.

Network Startup Script: /etc/rc.d/init.d/network and /etc/sysconfig/network-scripts

On Red Hat, your network interface is started up using the **network** script in the **/etc/rc.d/init.d** directory. This script will activate your network interface cards (NIC) as well as implement configuration information such as gateway, host, and name server identities. You can manually shut down and start your network interface using this script and the **restart**, **start**, or **stop** options. You can run the script on Red Hat with the **service** command. The following commands shut down and then start up your network interface:

```
service network stop
service network start
```

If you are changing network configuration, you will have to restart your network interface for the changes to take effect:

```
service network restart
```

To test if your interface is working, use the **ping** command with an IP address of a system on your network, such as your gateway machine. The **ping** command continually repeats until you stop it with a CTRL-C.

```
ping 192.168.0.1
```

The **/etc/rc.d/init.d/network** file performs the startup operations by executing several specialized scripts located in the **/etc/sysconfig/network-scripts** directory. The **network** script uses a script in that directory called **ifup** to activate a network connection, and **ifdown** to shut it down. **ifup** and **ifdown** will invoke other scripts tailored to the kind of device being worked on, such as **ifup-ppp** for modems using the PPP protocol, or **ifup-ipv6** for network devices that use IP Protocol version 6 addressing.

The **ifup** and **ifdown** scripts make use of interface configuration files that bear the names of the network interfaces currently configured such as **ifcfg-eth0** for the first Ethernet device. These files define shell variables that hold information on the interface, such as whether to start them at boot time. For example, the **ifcfg-eth0** file holds definitions for NETWORK, BROADCAST, and IPADDR, which are assigned the network, broadcast, and IP addresses that the device uses.

The **ifdown** and **ifup** scripts, in turn, hold the **ifconfig** and **route** commands to activate scripts using these variables defined in the interface configuration files. If you want to manually start up an interface with **ifup**, you simply use the interface configuration file as its argument. The following command starts up the second Ethernet card:

```
cd /etc/sysconfig/network-scripts
ifup ifcfg-eth1
```

Interface configuration files are automatically generated when you configure your network connections, say with netcfg, rp3, or Linuxconf. You can also manually edit these interface configuration files, making changes such as whether to start up the interface at boot or not (though using a configuration tool like netcfg is easier). A sample **ifcfg-eth0** file is shown here.

/etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
BOOTPROTO=static
BROADCAST=192.168.0.255
IPADDR=192.168.0.1
NETMASK=255.255.255.0
NETWORK=192.168.0.0
ONBOOT=yes
```

ifconfig

The **ifconfig** command takes as its arguments the name of an interface and an IP address, as well as options. The **ifconfig** command then assigns the IP address to the interface. Your system now knows that such an interface exists and that it references a particular IP address. In addition, you can specify whether the IP address is a host address or a network address. You can use a domain name for the IP address, provided the domain name is listed along with its IP address in the **/etc/hosts** file. The syntax for the **ifconfig** command is as follows:

```
# ifconfig interface -host_net_flag address options
```

The *host_net_flag* can be either **-host** or **-net** to indicate a host or network IP address. The **-host** flag is the default. The **ifconfig** command can have several options, which set different features of the interface, such as the maximum number of bytes it can transfer (**mtu**) or the broadcast address. The **up** and **down** options activate and deactivate the interface. In the next example, the **ifconfig** command configures an Ethernet interface:

```
# ifconfig eth0 192.168.0.1
```

For a simple configuration such as this, **ifconfig** automatically generates a standard broadcast address and netmask. The standard broadcast address is the network address with the number 255 for the host address. For a class C network, the standard netmask is 255.255.255.0, whereas for a class A network, the standard netmask is 255.0.0.0. If you are connected to a network with a particular netmask and broadcast address, however, you must specify them when you use **ifconfig**. The option for specifying the broadcast address is **broadcast;** for the network mask, it is **netmask**. Table 39-8 lists the different **ifconfig** options. In the next example, **ifconfig** includes the netmask and broadcast address:

```
# ifconfig eth0 192.168.0.1 broadcast 192.168.0.255 netmask 255.255.255.0
```

Table 39-8: The ifconfig Options

Option	Description
Interface	Name of the network interface, such as **eth0** for the first Ethernet device or **ppp0** for the first PPP device (modem)
aftype	Address family for decoding protocol addresses; default is inet, currently used by Linux
up	Activates an interface; implied if IP address is specified
down	Deactivates an interface
-arp	Turns ARP on or off; preceding **-**turns it off
-trailers	Turns on or off trailers in Ethernet frames; preceding **-** turns it off
-allmulti	Turns on or off the promiscuous mode; preceding **-** turns it off- this allows network monitoring
metric *n*	Cost for interface routing (not currently supported)
mtu *n*	Maximum number of bytes that can be sent on this interface per transmission
dstaddr *address*	Destination IP address on a point-to-point connection
netmask *address*	IP network mask; preceding **-** turns it off
broadcast *address*	Broadcast address; preceding **-** turns it off
point-to-point *address*	Point-to-point mode for interface; if address is included, it is assigned to remote system

Table 39-8: The ifconfig Options

Option	Description
hw	Sets hardware address of interface
Address	IP address assigned to interface

Once you configure your interface, you can use **ifconfig** with the **up** option to activate it and with the **down** option to deactivate it. If you specify an IP address in an **ifconfig** operation, as in the previous example, the **up** option is implied.

```
# ifconfig eth0 up
```

Point-to-point interfaces such as Parallel IP (PLIP), Serial Line IP (SLIP), and Point-to-Point Protocol (PPP) require you to include the **pointopoint** option. A PLIP interface name is identified with the name plip with an attached number. For example, **plip0** is the first PLIP interface. SLIP interfaces use **slip0**. PPP interfaces start with **ppp**0. Point-to-point interfaces are those that usually operate between only two hosts, such as two computers connected over a modem. When you specify the **pointopoint** option, you need to include the IP address of the host. In the next example, a PLIP interface is configured that connects the computer at IP address 192.168.1.72 with one at 204.166.254.14. If domain addresses were listed for these systems in **/etc/hosts**, those domain names could be used in place of the IP addresses.

```
# ifconfig plip0 192.168.1.72 pointopoint 204.166.254.14
```

If you need to, you can also use **ifconfig** to configure your loopback device. The name of the loopback device is **lo**, and its IP address is the special address 127.0.0.1. The following example shows the configuration:

```
# ifconfig lo 127.0.0.1
```

The **ifconfig** command is useful for checking on the status of an interface. If you enter the **ifconfig** command, along with the name of the interface, information about that interface is displayed:

```
# ifconfig eth0
```

To see if your loopback interface is configured, you can use **ifconfig** with the loopback interface name, **lo**:

```
# ifconfig lo

lo Link encap:Local Loopback
 inet addr:127.0.0.1 Bcast:127.255.255.255 Mask:255.0.0.0
 UP BROADCAST LOOPBACK RUNNING MTU:2000 Metric:1
 RX packets:0 errors:0 dropped:0 overruns:0
 TX packets:12 errors:0 dropped:0 overruns:0
```

Routing

A packet that is part of a transmission takes a certain *route* to reach its destination. On a large network, packets are transmitted from one computer to another until the destination computer is reached. The route determines where the process starts and to what computer your system needs to send the packet for it to reach its destination. On small networks, routing may be static—that is, the route from one system to another is fixed. One system knows how to reach another, moving through fixed paths. On larger networks and on the Internet, however, routing is dynamic. Your system knows the first computer to send its packet off to, and then that computer takes the packet from there, passing it on to another computer, which then determines where to pass it on. For dynamic routing, your system needs to know little. Static routing, however, can become complex because you have to keep track of all the network connections.

Your routes are listed in your routing table in the **/proc/net/route** file. To display the routing table, enter **route** with no arguments (the **netstat -r** command will also display the routing table):

```
# route
Kernel routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
loopback * 255.0.0.0 U 0 0 12 lo
pango1.train.com * 255.255.255.0 U 0 0 0 eth0
```

Each entry in the routing table has several fields, providing information such as the route destination and the type of interface used. The different fields are listed in the following table:

Field	Description
Destination	Destination IP address of the route
Gateway	IP address or hostname of the gateway the route uses; * indicates no gateway is used
Genmask	The netmask for the route
Flags	Type of route: U = up, H = host, G = gateway, D = dynamic, M = modified
Metric	Metric cost of route
Ref	Number of routes that depend on this one
Window	TCP window for AX.25 networks
Use	Number of times used
Iface	Type of interface this route uses

You should have at least one entry in the routing table for the loopback interface. If not, you must route the loopback interface using the **route** command. The IP address for an interface has to be added to the routing table before you can use that interface. You add an address with the **route** command and the **add** option:

```
route add address
```

The next example adds the IP address for the loopback interface to the routing table:

```
route add 127.0.0.1
```

With the **add** argument, you can add routes either for networks with the **–net** option or with the **–host** option for IP interfaces (hosts). The **–host** option is the default. In addition, you can then specify several parameters for information, such as the netmask (**netmask**), the gateway (**gw**), the interface device (**dev**), and the default route (**default**). If you have more than one IP interface on your system, such as several Ethernet cards, you must specify the name of the interface using the **dev** parameter. If your network has a gateway host, you use the **gw** parameter to specify it. If your system is connected to a network, at least one entry should be in your routing table that specifies the default route. This is the route taken by a message packet when no other route entry leads to its destination. The following example is the routing of an Ethernet interface:

```
# route add 192.168.1.2 dev eth0
```

If your system has only the single Ethernet device as your IP interface, you could leave out the **dev** eth0 parameter:

```
# route add 192.168.1.2
```

You can delete any route you establish by invoking **ifconfig** with the **del** argument and the IP address of that route, as in this example:

```
# route del 192.168.1.2
```

You also need to add routes for networks that an IP interface can access. For this, you use the **–net** option. In this example, a route is set up for a system's LAN at 192.168.1.0:

```
# route add -net 192.168.1.0 dev eth0
```

For a gateway, you first add a route to the gateway interface, and then add a route specifying that it is a gateway. The address of the gateway interface in this example is 192.168.1.1:

```
# route add 192.168.1.1
# route add default gw 192.168.1.1
```

If you are using the gateway to access a subnet, add the network address for that network (in this example, 192.168.23.0):

```
 # route add –net 192.168.23.0 gw dev eth1
```

To add another IP address to a different network interface on your system, use the **ifconfig** and **route** commands with the new IP address. The following command configures a second Ethernet card (eth1) with the IP address 192.168.1.3:

```
ifconfig eth1 192.168.1.3
route add 192.168.1.3 dev eth1
```

Monitoring Your Network: ping and netstat

With the ping program, you can check to see if you can actually access another host on your network. The ping program sends a request to the host for a reply. The host then sends a reply back, and it is displayed on your screen. The ping program continually sends such a request until you stop it with a **break** command, a CTRL-C. You see one reply after another scroll by on your screen until you stop the program. If ping cannot access a host, it issues a message saying the host is unreachable. If ping fails, this may be an indication that your network connection is not working. It may only be the particular interface, a basic configuration problem, or a bad physical connection. ping uses the Internet Control Message Protocol (ICMP) discussed in Chapter 40. Networks may block these protocols as a security measure, also preventing ping from working. A ping failure may simply indicate a security precaution on the part of the queried network.

To use ping, enter **ping** and the name of the host. You can also use the KDE network utilities on the KDE desktop and gfinger on the Gnome desktop (see Chapter 21).

```
$ ping ftp.redhat.com
```

The netstat program provides real-time information on the status of your network connections, as well as network statistics and the routing table. The netstat program has several options you can use to bring up different sorts of information about your network (see Table 38-9):

```
# netstat
Active Internet connections
Proto Recv-Q Send-Q Local Address Foreign Address (State) User
tcp 0 0 turtle.mytrek.com:01 pango1.mytrain.com.:ftp ESTABLISHED dylan
Active UNIX domain sockets
Proto RefCnt Flags Type State Path
unix 1 [ ACC ] SOCK_STREAM LISTENING /dev/printer
unix 2 [ ] SOCK_STREAM CONNECTED /dev/log
unix 1 [ ACC ] SOCK_STREAM LISTENING /dev/nwapi
unix 2 [ ] SOCK_STREAM CONNECTED /dev/log
unix 2 [ ] SOCK_STREAM CONNECTED
unix 1 [ ACC ] SOCK_STREAM LISTENING /dev/log
```

The **netstat** command with no options lists the network connections on your system. First, active TCP connections are listed, and then the active domain sockets are listed. The domain sockets contain processes used to set up communications among your system and other systems. The various fields are described in the following table. You can use **netstat** with the **-r** option to display the routing table, and **netstat** with the **-i** option displays the uses of the different network interfaces.

Table 39-9: The netstat Options

Option	Description
-a	Displays information about all Internet sockets, including those sockets that are only listening
-i	Displays statistics for all network devices
-c	Displays network status continually every second until the program is interrupted
-n	Displays remote and local addresses as IP addresses
-o	Displays timer states, expiration times, and backoff state for network connections
-r	Displays the kernel routing table
-t	Displays information about TCP sockets only, including those that are listening
-u	Displays information about UDP sockets only

Table 39-9: The netstat Options

Option	Description
-v	Displays version information
-w	Displays information about raw sockets only
-x	Displays information about Unix domain sockets

IP Aliasing

In some cases, you may want to assign a single Linux system that has only one network interface to two or more IP addresses. For example, you may want to run different Web sites that can be accessed with separate IP addresses on this same system. In effect, you are setting up an alias for your system, another address by which it can be accessed. In fact, you are assigning two IP addresses to the same network interface—for example, assigning a single Ethernet card two IP addresses. This procedure is referred to as *IP aliasing* and is used to set up multiple IP-based virtual hosts for Internet servers. This method enables you to run several Web servers on the same machine using a single interface (or more than one on each of several interfaces). See Chapters 23 and 24 for FTP and Web server information about virtual hosts, and Chapter 25 for Domain Name Service configuration.

Setting up an IP alias is a simple matter of configuring a network interface on your system to listen for the added IP address. Your system needs to know what IP addresses it should listen for and on what network interface. You set up IP aliases using either Linuxconf or the **ifconfig** and **route** commands. For Linuxconf, select the IP aliases for virtual hosts under Server tasks. This opens a panel that lists available interfaces. Click one to open a panel where you can enter added IP addresses for it.

To add another address to the same interface, you need to qualify the interface by adding a colon and a number. For example, if you are adding another IP address to the first Ethernet card (**eth0**), you would add a **:0** to its interface name, **eth0:0**. The following example shows the **ifconfig** and **route** commands for the Ethernet interface 192.168.1.2 and two IP aliases added to it: 192.168.1.100 and 192.168.1.101. To add yet another IP address to this same interface, you would use **eth0:1**, incrementing the qualifier, and so on. The first **ifconfig** command assigns the main IP address, 192.168.1.2, to the first Ethernet device, **eth0**. Then, two other IP addresses are assigned to that same device. In the first **route** command, the network route is set up for the Ethernet device, and then routes are set up for each IP interface. The interfaces for the two aliases are indicated with **eth0:0** and **eth0:1**:

```
ifconfig eth0 192.168.1.2
ifconfig eth0:0 192.168.1.100
ifconfig eth0:1 192.168.1.101
route add -net 192.168.1.0 dev eth0
route add -host 192.168.1.2 dev eth0
route add -host 192.168.1.100 dev eth0:0
route add -host 192.168.1.101 dev eth0:1
```

IP aliasing must be supported by the kernel before you can use it. If your kernel does not support it, you may have to rebuild the kernel (including IP aliasing support), or use loadable modules to add IP aliasing.

Chapter 40: Network Security: Firewalls, Encryption, and Authentication

Overview

Most systems currently connected to the Internet are open to attempts by outside users to gain unauthorized access. Outside users can try either to gain access directly by setting up an illegal connection, by intercepting valid communications from users remotely connected to the system, or by pretending to be a valid users. Firewalls, encryption, and authentication procedures are ways of protecting against such attacks. A *firewall* prevents any direct unauthorized attempts at access, *encryption* protects transmissions from authorized remote users, and *authentication* verifies that a user requesting access has the right to do so. The current Linux kernel incorporates support for firewalls using the Netfilter (iptables) packet filtering package (the previous version, ipchains, is used on older systems). To implement a firewall, you simply provide a series of rules to govern what kind of access you want to allow on your system. If that system is also a gateway for a private network, the system's firewall capability can effectively help protect the network from outside attacks. To provide protection for remote communications, transmission can be simply encrypted. For Linux systems, you can use the Secure Shell (SSH) suite of programs to encrypt any transmissions, preventing them from being read by anyone else. The SSH programs are meant to replace the remote tools such as rsh and rcp (see Chapter 21), which perform no encryption and include security risks such as transmitting passwords in clear text. In addition, Kerberos authentication provides another level of security whereby individual services can be protected, allowing use of a service only to users who are cleared for access. Table 40-1 lists several network security applications commonly used on Linux.

Table 40-1: Network Security Applications

Web Site	Security Application
netfilter.samba.org	Netfilter project, iptables, and NAT
netfilter.samba.org/ipchains	IP-Chains firewall
www.openssh.org	Secure Shell encryption
www.squid.org	Squid Web Proxy server
web.mit.edu/kerberos	Kerberos network authentication

Outside users may also try to gain unauthorized access through any Internet services you may be hosting, such as a Web site. In such a case, you can set up a proxy to protect your site from attack. For Linux systems, use Squid proxy software to set up a proxy to protect your Web server (see Chapter 28).

Firewalls: iptables and NAT

A good foundation for your network's security is to set up a Linux system to operate as a firewall for your network, protecting it from unauthorized access. You can use a firewall to implement either packet filtering or proxies. *Packet filtering* is simply the process of deciding whether a packet received by the firewall host should be passed on into the local network. The packet-filtering software checks the source and destination addresses of the packet and sends the packet on, if it's allowed. Even if your system is not part of a network but connects directly to the Internet, you can still use the firewall feature to control access to your system. Of course, this also provides you with much more security.

With proxies, you can control access to specific services, such as Web or FTP servers. You need a proxy for each service you want to control. The Web server has its own Web proxy, while an FTP server has an FTP proxy. Proxies can also be used to cache commonly used data, such as Web pages, so that users needn't constantly access the originating site. The proxy software commonly used on Linux systems is Squid, discussed in Chapter 27.

An additional task performed by firewalls is Network Address Translation (NAT). Network Address Translation redirects packets to appropriate destinations. It performs tasks such as redirecting packets to certain hosts, forwarding packets to other networks, and changing the host source of packets to implement IP masquerading.

Note ipchains was used to implement packet filtering and NAT tasks on firewalls for the Linux 2.2 kernel and earlier.

The Netfilter software package implements both packet filtering and NAT tasks for the Linux 2.4 kernel and above. The Netfilter software is developed by the Netfilter Project, which you can find out more about at **netfilter.samba.org**. The command used to execute packet filtering and NAT tasks is **iptables**, and the software is commonly referred to as simply iptables. However, Netfilter implements packet filtering and NAT tasks separately using different tables and commands. A table will hold the set of commands for its application. This approach streamlines the packet-filtering task, letting iptables perform packet-filtering checks without the overhead of also having to address translations. NAT operations are also freed from being mixed in with packet-filtering checks. You use the **iptables** command for both packet filtering and NAT tasks, but for NAT you add the **-nat** option.

The iptables software can be built directly into the 2.4 kernel or loaded as a kernel module, iptable_filter.o. Unlike its predecessor, ipchains, Netfilter is designed to be modularized and extensible. Capabilities can be added in the form of modules such as the state module, which adds connection tracking.

Iptables includes backward-compatible modules for both ipfwadm and ipchains. In fact, iptables is very similar to ipchains. You can still use ipchains and the earlier ipfwadm commands by loading the ipchains.o or ipfwadm.o modules provided with the Netfilter software. These provide full backward compatibility. For the sake of older systems, ipchains is also discussed here.

Packet Filtering

Netfilter is essentially a framework for packet management that can check packets for particular network protocols and notify parts of the kernel listening for them. Built on the Netfilter framework is the packet selection system implemented by IP Tables. With IP Tables, different tables of rules can be set up to select packets according to differing criteria. Packet filtering is implemented using a filter table that holds rules for dropping or accepting packets. Network Address Translation operations such as IP masquerading are implemented using the NAT table that holds IP masquerading rules. Preroute packet mangling uses the mangle table.

This structure is extensible in that new modules can define their own tables with their own rules. It also greatly improves efficiency. Instead of all packets checking one large table, they only access the table of rules they need to.

IP Tables rules are managed using the **iptables** command. For this command, you will need to specify the table you want to manage. The default is the filter table, which need not be specified. You can list the rules you have added at any time with the **–L** and **–n** options, as shown here. The **–n** option says to use only numeric output for both IP addresses and ports, avoiding a DNS lookup for hostnames. You could, however, just use the **–L** option to see the port labels and hostnames:

```
iptables –L -n
```

Note In iptables' commands, chain names have to be entered in uppercase. This means that the chain names INPUT, OUTPUT, and FORWARD have to be written in uppercase.

Packet-filtering rules are very similar to those used in ipchains with few exceptions. Rules are combined into different chains. The kernel uses chains to manage packets it receives and sends out. A *chain* is simply a checklist of rules. These rules specify what action to take for packets containing certain headers. The rules operate with an if-then-else structure. If a packet does not match the first rule, the next rule is then checked, and so on. If the packet does not match any rules, then the kernel consults chain policy. Usually, at this point the packet is rejected. If the packet does match a rule, it is passed to its target, which determines what to do with the packet. The standard targets are listed in Table 40-2. If a packet does not match any of the rules, it is passed to the chain's default target.

Table 40-2: iptables Targets

Target	**Function**
ACCEPT	Allows packet to pass through the firewall
DROP	Denies access by the packet (same as DENY in ipchains)
REJECT	Denies access and notifies the sender
QUEUE	Sends packets to user space
RETURN	Jumps to the end of the chain and lets the default target process it

A *target* could, in turn, be another chain of rules, even a chain of user-defined rules. A packet could be passed through several chains before finally reaching a target. In the case of user-defined chains, the default target is always the next rule in the chains from which it was called. This sets up a procedure or function call—like flow of control found in programming languages. When a rule has a user-defined chain as its target, then, when activated, that user-defined chain is executed. If no rules are matched, execution returns to the next rule in the originating chain.

The kernel uses three firewall chains: INPUT, OUTPUT, and FORWARD. When a packet is received through an interface, the INPUT chain is used to determine what to do with it. The kernel then uses its routing information to decide where to send it. If the kernel sends the packet to another host, the FORWARD chain is checked. Before the packet is actually sent, the OUTPUT chain is also checked. In addition, two NAT table chains, POSTROUTING and PREROUTING, are implemented to handle masquerading and packet address modifications. The built-in Netfilter chains are listed in Table 40-3.

Table 40-3: Netfilter Built-in Chains

Chain	**Descriptions**
INPUT	Rules for incoming packets
OUTPUT	Rules for outgoing packets
FORWARD	Rules for forwarded packets
PREROUTING	Rules for redirecting or modifying incoming packets, NAT table only
POSTROUTING	Rules for redirecting or modifying outgoing packets, NAT table only

You add and modify chain rules using the **iptables** commands. An **iptables** command consists of the keyword **iptables**, followed by an argument denoting the command to execute. For example, **iptables -A** is the command to add a new rule, whereas **iptables -D** is the command to delete a rule. The **iptables** commands are listed in Table 40-4. The following command simply lists the chains along with their rules currently defined for your system. The output shows the default values created by **iptables** commands.

Table 40-4: iptables Commands

Option	**Function**
-A *chain*	Appends a rule to a chain
-D *chain*	Deletes matching rule from a chain
-D *chain rulenum*	Deletes rule *rulenum* (1 = first) from *chain*
-I *chain* **[***rulenum***]**	Inserts in *chain* as *rulenum* (default 1 = first)
-R *chain rulenum*	Replaces rule *rulenum* (1 = first) in *chain*
-L [*chain***]**	Lists the rules in *chain* or all chains
-E [*chain***]**	Renames a chain
-F [*chain***]**	Deletes (flush) all rules in *chain* or all chains
-R *chain*	Replaces a rule; rules are numbered from 1
-Z [*chain***]**	Zero counters in *chain* or all chains
-N *chain*	Create a new user-defined chain
-X *chain*	Deletes a user-defined chain
-P *chain target*	Changes policy on *chain* to *target*

```
iptables –L -n
Chain input (policy ACCEPT):
Chain forward (policy ACCEPT):
Chain output (policy ACCEPT):
```

To add a new rule to a chain, you use **-A**. Use **-D** to remove it, and **-R** to replace it. Following the command, list the chain to which the rule applies, such as the INPUT, OUTPUT, or FORWARD chain, or else a user-defined chain. Next, you list different options that specify the actions you want taken. Most are the same as those used for ipchains, with a few exceptions. The **-s** option specifies the source address attached to the packet, **-d** specifies the destination address, and **-j** specifies the target. The ACCEPT target will allow a packet to pass. The **-i** option now indicates the input device and can only be used with the INPUT and

FORWARD chains. The **-o** option indicates the output device and can only be used for OUTPUT and FORWARD chains. Table 40-5 lists several basic options. Many are similar to the ipchains options, but some, like **-i**, are different and others are missing (like **–y)**.

Table 40-5: iptables Options

Option	**Function**
-p [!] *proto*	Specify a protocol, such as TCP, UDP, ICMP, or ALL.
-s [!] *address***[/***mask***] [!] [***port***[:***port***]]**	Source address to match. With the *port* argument, you can specify the port.
--sport [!] [*port***[:***port***]]**	Source port specification. You can specify a range of ports using the colon, *port:port*.
-d [!] *address***[/***mask***] [!]** [*port***[:***port***]]**	Destination address to match. With the *port* argument, you can specify the port.
--dport [!][*port***[:***port***]]**	Destination port specification.
--icmp-type [!] *typename*	Specify ICMP type.
-i [!] *name*[+]	Specify an input network interface using its name (for example, **eth0**). The + symbol functions as a wildcard. The + attached to the end of the name matches all interfaces with that prefix (**eth+** matches all Ethernet interfaces). Can only be used with the INPUT chain.
-j *target* **[port]**	Specify the target for a rule (specify **[port]** for REDIRECT target).
--to-source *ipaddr***[-** *ipaddr***]** **[:** *port***-** *port***]**	Used with the SNAT target, rewrites packets with new source IP address.
--to-destination *ipaddr* **[-** *ipaddr***][:** *port***-** *port***]**	Used with the DNAT target, rewrites packets with new destination IP address.
-n	Numeric output of addresses and ports, used with **–L**.
-o [!] *name*[+]	Specify an output network interface using its name (for example, **eth0**). Can only be used with FORWARD and OUTPUT chains.
-t *table*	Specify a table to use, as in **–t nat** for the NAT table.
-v	Verbose mode, shows rule details, used with **–L**.
-x	Expand numbers (display exact values), used with **–L**.
[!] –f	Match second through last fragments of a fragmented packet.
[!] –V	Print package version.
!	Negates an option or address.
-m	Specify a module to use, such as state.
--state	Specify options for the state module such as NEW, INVALID, RELATED, and ESTABLISHED. Used to detect packets state. NEW references SYN packets (new connections).
--syn	SYN packets, new connections.

Table 40-5: iptables Options

Option	Function
--tcp-flags	Tcp flags: SYN, ACK, FIN, RST, URG, PS, and ALL for all flags.
--limit	Option for the limit module (**-m limit**). Used to control the rate of matches, matching a given number of times per second.
--limit-burst	Option for the limit module (**-m limit**). Specify maximum burst before the limit kicks in. Used to control denial of service attacks.

Iptables is designed to be extensible, and there are number of options with selection criteria that can be included with iptables. For example, the TCP extension includes the **--syn** option that checks for SYN packets. The ICMP extension provides the **--icmp- type** option for specifying ICMP packets as those used in ping operations. The limit extension includes the **--limit** option with which you can limit the maximum number of matching packets in a specified time period, like a second.

In the following example, the user adds a rule to the INPUT chain to accept all packets originating from the address 192.168.0.55. Any packets that are received (**INPUT**) whose source address (**-s**) matches 192.168.0.55 are accepted and passed through (**-j ACCEPT**).

```
iptables -A INPUT -s 192.168.0.55 -j ACCEPT
```

There are two built-in targets, DROP and ACCEPT. DROP is the same as the ipchains DENY target. Other targets can be either user-defined chains or extensions added on, such as REJECT. There are two special targets used to manage chains, RETURN and QUEUE. RETURN indicates the end of a chain and returns to the chain it started from. QUEUE is used to send packets to userspace (replaces **-o** in ipchains).

```
iptables -A INPUT -s www.myjunk.com -j DROP
```

You can turn a rule into its inverse with a **!** symbol. For example, to accept all incoming packets except those from a specific address, place a **!** symbol before the **-s** option and that address. The following example will accept all packets except those from the IP address 192.168.0.45:

```
iptables -A INPUT -j ACCEPT ! -s 192.168.0.45
```

You can specify an individual address using its domain name or its IP number. For a range of addresses, you can use the IP number of their network and the network IP mask. The IP mask can be an IP number or simply the number of bits making up the mask. For example, all of the addresses in network 192.168.0 can be represented by 192.168.0.0/225.255.255.0 or by 192.168.0.0/24. To specify any address, you can use 0.0.0.0/0.0.0.0 or simply 0/0. By default, rules reference any address if no **-s** or **-d** specification exists. The following example accepts messages coming in that are from (source) any host in the 192.168.0.0 network and that are going (destination) anywhere at all (the **-d** option is left out or could be written as **-d 0/0**):

```
iptables -A INPUT -s 192.168.0.0/24  -j ACCEPT
```

The iptables rules are usually applied to a specific network interface such as the Ethernet interface used to connect to the Internet. For a single system connected to the Internet, you will have two interfaces, one that is your Internet connection and a localhost interface (**lo**) for internal connections between users on your system. The network interface for the Internet is referenced using the device name for the interface. For example, an Ethernet card with the device name **/dev/eth0** would be referenced by the name **eth0**. A modem using PPP protocols with the device name **/dev/ppp0** would have the name **ppp0**. In iptables rules, you use **-i** option to indicate the input device; it can only be used with the INPUT and FORWARD chains. The **-o** option indicates the output device and can only be used for OUTPUT and FORWARD chains. Rules can then be applied to packets arriving and leaving on particular network devices. In the following examples, the first rule references the Ethernet device **eth0**, and the second, the localhost:

```
iptables -A INPUT -j DROP -i eth0 -s 192.168.0.45
iptables -A INPUT -j ACCEPT  -i lo
```

User Defined Chains

With iptables, the FORWARD and INPUT chains are evaluated separately. One does not feed into the other. This means that if you want to completely block certain addresses from passing through your system, you will need to add both a FORWARD and INPUT rule for them.

```
iptables -A INPUT -j DROP -i eth0 -s 192.168.0.45
iptables -A FORWARD -j DROP -i eth0 -s 192.168.0.45
```

A common method for reducing repeated INPUT and FORWARD rules is to create a user chain that both the INPUT and FORWARD chains feed into. You define a user chain with the **–N** option. The next example shows the basic format for this arrangement. A new chain is created called incoming (it can be any name you choose). The rules you would define for your FORWARD and INPUT chains are now defined for the incoming chain. The INPUT and FORWARD chains then use the incoming chain as a target, jumping directly to it and using its rules to process any packets they receive.

```
iptables –N incoming

iptables -A incoming -j DROP -i eth0 -s 192.168.0.45
iptables -A incoming -j ACCEPT  -i lo

iptables -A FORWARD -j incoming
iptables –A INPUT –j incoming
```

ICMP Packets

Firewalls often block certain Internet Control Message Protocol (ICMP) messages. ICMP redirect messages, in particular, can take control of your routing tasks. You need to enable some ICMP messages, however, such as those needed for ping, traceroute, and particularly destination-unreachable operation. In most cases, you always need to make sure destination-unreachable packets are allowed; otherwise, domain name queries could hang. Some of the more common ICMP packet types are listed in Table 40-6. You can enable an ICMP type of packet with the **-icmp-type** option, which takes as its argument a number or a name representing the message. The following examples enable the use of echo-reply, echo-request, and destination-unreachable messages, which have the numbers 0, 8, and 3.

```
iptables -A INPUT -j ACCEPT  -p icmp -i eth0 --icmp-type  echo-reply -d
10.0.0.1
iptables -A INPUT -j ACCEPT  -p icmp -i eth0 --icmp-type  echo-request -d
10.0.0.1
iptables -A INPUT -j ACCEPT -p icmp -i eth0 --icmp-type  destination-
unreachable -d 10.0.0.1
```

Table 40-6: Common ICMP Packets

Number	Name	Required by
0	echo-reply	ping
3	destination-unreachable	Any TCP/UDP traffic
5	redirect	Routing if not running routing daemon
8	echo-request	ping
11	time-exceeded	traceroute

Their rule listing will look like this:

```
ACCEPT     icmp --  0.0.0.0/0            10.0.0.1           icmp type 0
ACCEPT     icmp --  0.0.0.0/0            10.0.0.1           icmp type 8
ACCEPT     icmp --  0.0.0.0/0            10.0.0.1           icmp type 3
```

Ping operations need to be further controlled to avoid the ping of death security threat. You can do this several ways. One way is to deny any ping fragments. Ping packets are normally very small. You can block ping of death attacks by denying any ICMP packet that is a fragment. Use the **–f** option to indicate fragments.

```
iptables –A INPUT –p icmp –j DROP –f
```

Another way is to limit the number of matches received for ping packets. You use the limit module to control the number of matches on the ICMP ping operation. Use **–m limit** to use the limit module, and **–-limit** to specify the number of allowed matches. **1/s** will allow one match per second.

```
iptables -A FORWARD -p icmp --icmp-type echo-request -m limit --limit
 1/s -j ACCEPT
```

Ports

If your system is hosting an Internet service, such as a Web or FTP server, you can use iptables to control access to it. You can specify a particular service by using the source port (**--sport**) or destination port (**--dport**) options with the port that the service uses. iptables lets you use names for ports such as **www** for the Web server port. The names of services and the ports they use are listed in the **/etc/services** file, which maps ports to particular services. For a domain name server, the port would be **domain**. You can also use the port number if you want, preceding the number with a colon. The following example accepts all messages to the Web server located at 192.168.0.43:

```
iptables -A INPUT -d 192.168.0.43 --dport www -j ACCEPT
```

You can also use port references to protect certain services and deny others. This approach is often used if you are designing a firewall that is much more open to the Internet, letting users make freer use of Internet connections. Certain services you know can be harmful, such as telnet and ntp, can be denied selectively. For example, to deny any kind of telnet operation on your firewall, you can drop all packets coming in on the telnet port, 23. To protect NFS operations, you can deny access to the port used for the portmapper, 111. You can use either the port number or the port name.

```
# deny outside access to portmapper port on firewall.
iptables -A arriving  -j DROP -p tcp -i eth0  --dport 111
# deny outside access to telnet port on firewall.
iptables -A arriving  -j DROP -p tcp -i eth0  --dport telnet
```

The rule listing will look like this:

```
DROP      tcp  --  0.0.0.0/0    0.0.0.0/0     tcp dpt:111
DROP      tcp  --  0.0.0.0/0    0.0.0.0/0     tcp dpt:23
```

One port-related security problem is the access to your X server on the XFree86 ports that range from 6000 to 6009. On a relatively open firewall, these ports could be used to illegally access your system through your X server. A range of ports can be specified with a colon, as in 6000:6009. You can also use x11 for the first port, as in x11:6009. Sessions on the X server can be secured by using ssh, which normally accesses the X server on port 6010.

```
iptables -A arriving  -j DROP -p tcp -i eth0  --dport 6000:6009
```

Common ports checked and their labels are shown here:

Service	**Port Number**	**Port Label**
Auth	113	auth
Finger	79	finger
FTP	21	ftp
NTP	123	ntp
Portmapper	111	sunrpc
Telnet	23	telnet
Web server	80	www
XFree86	6000:6009	x11:6009

States

One of the more useful extensions is the state extension, which can easily detect tracking information for a packet. You need to specify the state module first with **-m state**. Then you can use the **--state** option. Here you can specify any of the following states:

State	**Description**
NEW	A packet that creates a new connection
ESTABLISHED	A packet that belongs to an existing connection
RELATED	A packet that is related to, but not part of, an existing connection, such as an ICMP error or a packet establishing an

State	Description
	FTP data connection
INVALID	A packet that could not be identified for some reason
RELATED+REPLY	A packet that is related to an established connection, but not part of one directly

If you are designing a firewall that is meant to protect your local network from any attempts to penetrate it from an outside network, you may want to restrict packets coming in. Simply denying access by all packets is unfeasible because users connected to outside servers—say, on the Internet—must receive information from them. You can, instead, deny access by a particular kind of packet used to initiate a connection. The idea is that an attacker must initiate a connection from the outside. The headers of these kinds of packets have their SYN bit set on and their FIN and ACK bits empty. The state module's NEW state matches on any such SYN packet. By specifying a DROP target for such packets, you deny access by any packet that is part of an attempt to make a connection with your system. Anyone trying to connect to your system from the outside is unable to do so. Users on your local system who have initiated connections with outside hosts can still communicate with them. The following example will drop any packets trying to create a new connection on the **eth0** interface, though they will be accepted on any other interface:

```
iptables -A INPUT -m state --state NEW -i eth0 -j DROP
```

You can use the **!** operator on the **eth0** device combined with an ACCEPT target to compose a rule that will accept any new packets except those on the **eth0** device. If the **eth0** device is the only one that connects to the Internet, this still effectively blocks outside access. At the same time, input operation for other devices such as your localhost are free to make new connections. This kind of conditional INPUT rule is used to allow access overall with exceptions. It usually assumes that a later rule such as a chain policy will drop remaining packets.

```
iptables -A INPUT -m state --state NEW ! -i eth0 -j ACCEPT
```

The next example will accept any packets that are part of an established connection or related to such a connection on the **eth0** interface:

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Network Address Translation (NAT)

Network Address Translation (NAT) is the process whereby a system will change the destination or source of packets as they pass through the system. A packet will traverse several linked systems on a network before it reaches its final destination. Normally, they will simply pass the packet on. However, if one of these systems performs a NAT on a packet, it can change the source or destination. A packet sent to a particular destination could have its destination address changed. To make this work, the system also needs to remember such changes so that the source and destination for any reply packets are altered back to the original addresses of the packet being replied to.

NAT is often used to provide access to systems that may be connected to the Internet through only one IP address. Such is the case with networking features like IP masquerading, support for multiple servers, and transparent proxying. With IP masquerading, NAT operations will change the destination and source of a packet moving through a firewall/gateway linking the Internet to computers on a local network. The gateway has a single IP address that the other local computers can use through NAT operations. If you have multiple servers but only one IP address, you can use NAT operations to send packets to the alternate servers. You can also use NAT operations to have your IP address reference a particular server application such as a Web server (transparent proxy).

Packet selection rules for NAT operations are added to the NAT table managed by the **iptables** command. To add rules to the NAT table, you have to specify the NAT table with the **-t** option. Thus to add a rule to the NAT table, you would have to specify the NAT table with the **-t nat** option as shown here:

```
iptables -t nat
```

With the **–L** option, you can list the rules you have added to the NAT table:

```
iptables -t nat –L -n
```

Adding the **–n** option will list IP addresses and ports in numeric form. This will speed up the listing as iptables will not attempt to do a DNS lookup to determine the hostname for the IP address.

In addition, there are two types of NAT operations: source NAT, specified as SNAT target, and destination NAT, specified as DNAT target. SNAT target is used for rules that alter source addresses, and DNAT target for those that alter destination addresses.

Three chains in the NAT table are used by the kernel for NAT operations. These are PREROUTING, POSTROUTING, and OUTPUT. PREROUTING is used for destination NAT (DNAT) rules. These are packets that are arriving. POSTROUTING is used for source NAT (SNAT) rules. These are for packets leaving. OUTPUT is used for destination NAT rules for locally generated packets.

As with packet filtering, you can specify source (**-s**) and destination (**-d**) addresses, as well as the input (**-i**) and output (**-o**) devices. The **-j** option will specify a target such as MASQUERADE. You would implement IP masquerading by adding a MASQUERADE rule to the POSTROUTING chain:

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

To change the source address of a packet leaving your system, you would use the POSTROUTING rule with the SNAT target. For the SNAT target, you use the **—to-source** option to specify the source address:

```
# iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 192.168.0.4
```

To change the destination address of packets arriving on your system, you would use the PREROUTING rule with the DNAT target and the **--to-destination** option:

```
# iptables -t nat -A PRETROUTING -i eth0 \
             -j DNAT --to-destination 192.168.0.3
```

To redirect an packet, you use the REDIRECT target on the PREROUTING chain:

```
# iptables -t nat -A PREROUTING -i eth1 --dport 80 -j REDIRECT --to-port
3128
```

With the TOS and MARK targets, you can mangle the packet to control its routing or priority. A TOS target sets the type of service for a packet, which can set the priority using criteria such as normal-service, minimize-cost, and maximize-throughput, among others.

The targets only valid for the NAT table are shown here:

SNAT	Modify source address; use **--to-source** option to specify new source address.
DNAT	Modify destination address; use **--to-destination** option to specify new destination address
REDIRECT	Redirect a packet.
MASQUERADE	IP masquerading.
MIRROR	Reverse source and destination and send back to sender.
MARK	Modify the Mark field to control message routing.
TOS	Modify the Type of Service field to manage the priority of the packet.

IP Tables Scripts

Though you can enter iptables rules from the shell command line, when you shut down your system, these commands will be lost. On Red Hat, you can make use of the built-in support for saving and reading iptables rules using the iptables service script. Alternatively, you can manage the process yourself, saving to files of your own choosing. In either event, you will most likely need to place your iptables rules in a script that can then be executed directly. This way you can edit and manage a complex set of rules, adding comments and maintaining their ordering.

Red Hat iptables Support

Red Hat provides support for iptables as part of its system configuration. When you install the Red Hat package for iptables, an iptables service script is installed that will read and save iptables commands using the **/etc/sysconfig/iptables** file. If you have set iptables to be started up automatically when you boot your system, this file will be checked to see if it exists and is not empty. If so, then iptables will automatically read the iptables commands that it holds. Red Hat is attempting to integrate iptables more smoothly into the system setup process.

You can sidestep this automatic iptables setup by simply deleting the **/etc/ sysconfig/iptables** file (running lokkit and choosing No Firewall will do the same). Be sure you back it up first in case it has important commands.

It is possible to edit the **/etc/sysconfig/iptables** file directly and enter iptables commands, but it is not recommended. Red Hat adds some notation of its own, such as a colon at the beginning of each line, and uses the notation to detect commands. Instead, you should think of this file as holding a final installation of your iptables commands.

You should think of the iptables service script that Red Hat provides as a versatile management tool, not as a service startup script. The use of the **service** command for this script can be confusing. The iptables script only manages iptables rules, flushing, adding, or reporting them. It does not start and stop the iptables service. If Netfilter is not running, you will need to instruct that it be started up when your system boots. For this, you can use setuptool (setup or Text Mode Setup Tool) to select System Services (ntsysv), then select ipchains from the list of services (press SPACEBAR). Make sure that ipchains is not selected. ipchains and iptables cannot run at the same time.

The service script **/etc/rc.d/init.d/iptables** supports several options with which to manage your rules. The **status** option displays a listing of all your current rules. The **stop** option will flush your current rules. Unlike **stop** and **status**, the **start** and **save** options are tied directly to the **/etc/sysconfig/iptables** file. The **start** option will flush your current iptable rules and add those in the **/etc/sysconfig/iptables** file. The **save** option will save your current rules to the **/etc/sysconfig/iptables** file. Keep in mind that the **stop** and **status** operations work on the current iptables rules, no matter if they were added manually on the command line, added by your own script, or added by the **start** option from **/etc/sysconfig/iptables**. The following command will list your current rules:

```
service iptables status
```

Perhaps the most effective way to think of the iptables service script is as an iptables development tool. When creating firewall rules, you should first create a script and place your rules in them, as described later on in the iptables script example. Make the script executable. Any changes you need to make as you debug your firewall, you make to this script. Before you run it, run the iptables service script with the **stop** option to clear out any previous rules:

```
service iptables stop
```

Then run your script, as shown here for the myfilters script:

```
./myfilters
```

To see how the commands have been interpreted by iptables, use the service script with the status option:

```
service iptables status
```

For any changes, edit your iptables script. Then run the service script again to clear out the old rules. Run the iptables script again, and use the **status** option with the service script to see how the commands were implemented:

```
service iptables stop
./myfilters
service iptables status
```

Once you are satisfied that your iptables rules are working correctly, you can save your rules to the **/etc/sysconfig/iptables** file. Use the iptables service script with the **save** option. Now your rules will be read automatically when your system starts up. You can think of the save operation as installing your iptables rules on your system, making them part of your system setup whenever you start your system.

```
service iptables save
```

To make changes, modify your iptables script, run the service script with **stop** to clear out the old rules, run the iptables script, and then use the service script with the **save** option to generate a new **/etc/sysconfig/iptables** file.

Manually Saving and Reading Rules

Instead of using the service script, you can save your rules to a file of your choosing using the **iptables-save** script. The recommended file to use is **/etc/iptables.rules**. The **iptables-save** command outputs rules to the standard output. To save them in a file, you must redirect the output to a file with the redirection operator, >, as shown here:

```
iptables-save > /etc/iptables.rules
```

Then, to restore the rules, use the **iptables-restore** script to read the **iptables** commands from that saved file:

```
iptables-restore < /etc/iptables.rules
```

You could then place the **iptables-restore** operation in the **/etc/rc.d/rc.local** script to have it run automatically.

An iptables Script Example

You now have enough information to create a simple iptables script that will provide basic protection for a single system connected to the Internet. The following script provides an IP Tables filtering process to protect a local network and a Web site from outside attacks. It configures a simple firewall for a private network (check the ipchains HOWTO for a more complex example). If you have a local network, you could adapt this script to it. In this configuration, all remote access initiated from the outside is blocked, but two-way communication is allowed for connections that users in the network make with outside systems. In this example, the firewall system functions as a gateway for a private network whose network address is 192.168.0.0 (see Figure 40-1). The Internet address is, for the sake of this example, 10.0.0.1. The system has two Ethernet devices: one for the private network (**eth1**) and one for the Internet (**eth0**). The gateway firewall system also supports a Web server at address 10.0.0.2. Entries in this example that are too large to fit on one line are continued on a second line, with the newline quoted with a backslash.

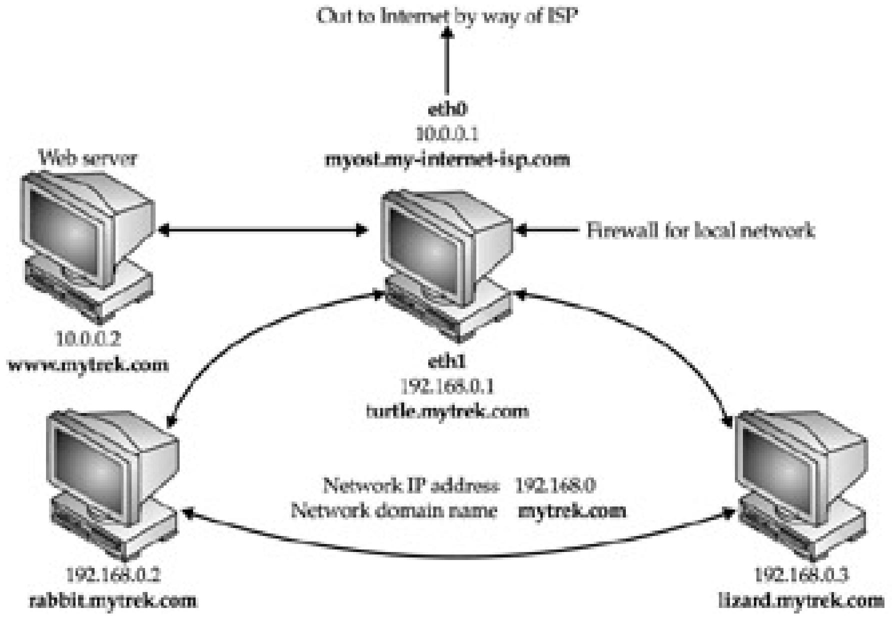

Out to Internet by way of ISP
eth0
10.0.0.1
myost.my-internet-isp.com
Web server
Firewall for local network
10.0.0.2
www.mytrek.com
eth1
192.168.0.1
turtle.mytrek.com
Network IP address 192.168.0
Network domain name mytrek.com
192.168.0.2
rabbit.mytrek.com
192.168.0.3
lizard.mytrek.com

Figure 40-1: A network with a firewall

The basic rules as they apply to different parts of the network are illustrated in Figure 40-2.

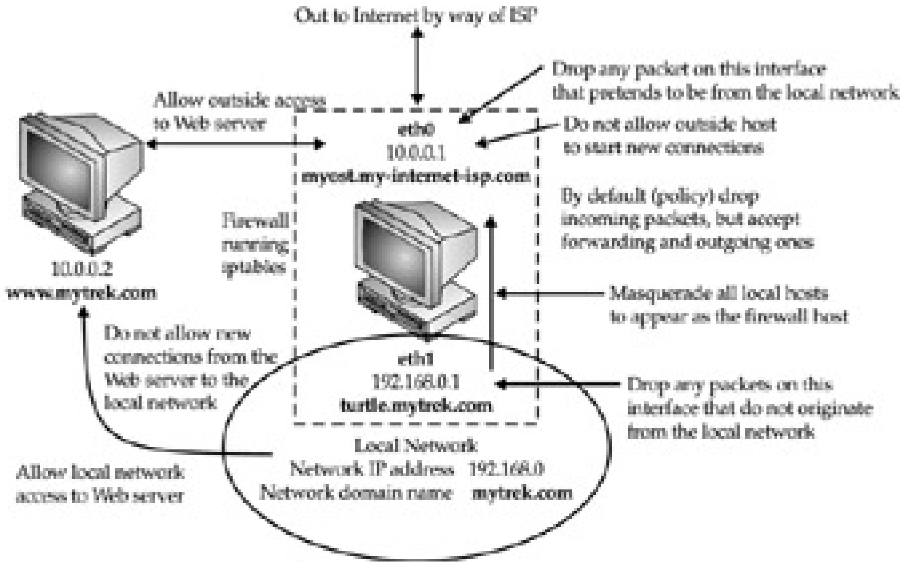

Out to Internet by way of ISP
Drop any packet on this interface that pretends to be from the local network
Allow outside access to Web server
eth0
10.0.0.1
myost.my-internet-isp.com
Do not allow outside host to start new connections
By default (policy) drop incoming packets, but accept forwarding and outgoing ones
Firewall running iptables
10.0.0.2
www.mytrek.com
Masquerade all local hosts to appear as the firewall host
Do not allow new connections from the Web server to the local network
eth1
192.168.0.1
turtle.mytrek.com
Drop any packets on this interface that do not originate from the local network
Local Network
Network IP address 192.168.0
Network domain name mytrek.com
Allow local network access to Web server

Figure 40-2: Firewall rules applied to a local network example

First a DROP policy is set up for INPUT and FORWARD built-in IP chains. This means that if a packet does not meet a criterion in any of the rules to let it pass, it will be dropped. Then both IP spoofing attacks and any attempts from the outside to initiate connections (SYN packets) are rejected. Outside connection attempts are also logged. This is a very basic configuration that can easily be refined to your own needs by adding IP Tables rules.

myfilter

```
# Firewall Gateway system IP address is 10.0.0.1 using Ethernet device eth0
# Private network address is 192.168.0.0 using Ethernet device eth1
# Web site address is 10.0.0.2

# modprobe iptable_filter
# turn off IP forwarding
```

```
echo 0 > /proc/sys/net/ipv4/ip_forward

# Flush chain rules
iptables -F INPUT
iptables -F OUTPUT
iptables -F FORWARD

# set default (policy) rules
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT

# IP spoofing, deny any packets on the internal network that have an
external
source address.
iptables -A INPUT -j LOG  -i eth1 \! -s 192.168.0.0/24
iptables -A INPUT -j DROP  -i eth1 \! -s 192.168.0.0/24
iptables -A FORWARD -j DROP  -i eth1 \! -s 192.168.0.0/24
# IP spoofing, deny any outside packets (any not on eth1) that have the
source
address of the internal network
iptables -A INPUT -j DROP \! -i eth1 -s 192.168.0.0/24
iptables -A FORWARD -j DROP \! -i eth1 -s 192.168.0.0/24
# IP spoofing, deny any outside packets with localhost address
# (packets not on the lo interface (any on eth0 or eth1) that have the
source
address of localhost)
iptables -A INPUT -j DROP  -i \! lo  -s  127.0.0.0/255.0.0.0
iptables -A FORWARD -j DROP  -i \! lo  -s  127.0.0.0/255.0.0.0

# allow all incoming messages for users on your firewall system
iptables -A INPUT -j ACCEPT  -i lo

# allow  communication to the Web server (address 10.0.0.2), port www
iptables -A INPUT  -j ACCEPT -p tcp -i eth0  —dport www -s 10.0.0.2
# Allow  established connections from Web servers to internal network
iptables -A INPUT -m state —state ESTABLISHED,RELATED -i eth0 -p tcp  —
sport www
-s 10.0.0.2 -d 192.168.0.0/24  -j ACCEPT
# Prevent new  connections from Web servers to internal network
iptables -A OUTPUT -m state —state  NEW -o eth0 -p tcp  —sport www -d
192.168.0.0/24  -j DROP

# allow established and related outside communication to your system
# allow outside communication to the firewall,except for ICMP packets
iptables -A INPUT -m state —state ESTABLISHED,RELATED -i eth0 -p \! icmp -j
ACCEPT
# prevent outside initiated connections
iptables -A INPUT -m state —state NEW -i eth0 -j DROP
iptables -A FORWARD -m state —state NEW -i eth0 -j DROP

# allow all local communication to and from the firewall on eth1  from the
local network
iptables -A INPUT -j ACCEPT -p all -i eth1 -s 192.168.0.0/24

# Set up masquerading to allow internal machines access to outside network
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

# Accept ICMP Ping (0 and 8) and Destination unreachable (3)  messages
# Others will be rejected by INPUT and OUTPUT DROP policy
```

```
iptables -A INPUT -j ACCEPT  -p icmp -i eth0 —icmp-type  echo-reply -d
10.0.0.1
iptables -A INPUT -j ACCEPT  -p icmp -i eth0 —icmp-type  echo-request -d
10.0.0.1
iptables -A INPUT -j ACCEPT -p icmp -i eth0 —icmp-type  destination-
unreachable -d 10.0.0.1

 # Turn on IP Forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Initially, in the script you would clear your current iptables with the flush option (**-F**), and then set the policies (default targets) for the non-user-defined rules. IP forwarding should also be turned off while the chain rules are being set:

```
echo 0 > /proc/sys/net/ipv4/ip_forward
```

One way to protect the private network from IP spoofing any packets is to check for any outside addresses on the Ethernet device dedicated to the private network. In this example, any packet on device **eth1** (dedicated to the private network) whose source address is not that of the private network (**! -s 192.168.0.0**) is denied. Also, check to see if any packets coming from the outside are designating the private network as their source. In this example, any packets with the source address of the private network on any Ethernet device other than for the private network (**eth1**) are denied. The same strategy can be applied to the local host.

```
# IP spoofing, deny any packets on the internal network
# that has an external source address.
iptables -A INPUT -j LOG  -i eth1 \! -s 192.168.0.0/24
iptables -A INPUT -j DROP  -i eth1 \! -s 192.168.0.0/24
iptables -A FORWARD -j DROP  -i eth1 \! -s 192.168.0.0/24
# IP spoofing, deny any outside packets (any not on eth1)
# that have the source address of the internal network
iptables -A INPUT -j DROP \! -i eth1 -s 192.168.0.0/24
iptables -A FORWARD -j DROP \! -i eth1 -s 192.168.0.0/24
# IP spoofing, deny any outside packets with localhost address
# (packets not on the lo interface (any on eth0 or eth1)
# that have the source address of localhost)
iptables -A INPUT -j DROP  -i \! lo  -s  127.0.0.0/255.0.0.0
iptables -A FORWARD -j DROP  -i \! lo  -s  127.0.0.0/255.0.0.0
```

Then, you would set up rules to allow all packets sent and received within your system (localhost) to pass:

```
iptables -A INPUT -j ACCEPT  -i lo
```

For the Web server, you want to allow access by outside users, but block access by anyone attempting to initiate a connection from the Web server into the private network. In the next example, all messages are accepted to the Web server, but the Web server cannot initiate contact with the private network. This prevents anyone from breaking into the local network through the Web server, which is open to outside access. Established connections are allowed, permitting the private network to use the Web server.

```
# allow  communication to the Web server (address 10.0.0.2), port www
iptables -A INPUT  -j ACCEPT -p tcp -i eth0  --dport www -s 10.0.0.2
```

```
# Allow  established connections from Web servers to internal network
iptables -A INPUT -m state --state ESTABLISHED,RELATED -i eth0 \
     -p tcp  --sport www -s 10.0.0.2 -d 192.168.0.0/24  -j ACCEPT
# Prevent new  connections from Web servers to internal network
iptables -A OUTPUT -m state --state  NEW -o eth0 -p tcp \
    --sport www -d 192.168.0.1.0/24  -j DROP
```

To allow access by the firewall to outside networks, you allow input by all packets except for ICMP packets. These are handled later. The firewall is specified by the firewall device, **eth0**. First, allow established and related connections to proceed:

```
# allow outside communication to the firewall,
# except for ICMP packets
iptables -A INPUT -m state --state ESTABLISHED,RELATED \
         -i eth0 -p \! icmp -j ACCEPT
```

To prevent outsiders from initiating any access to your system, create a rule to block access by SYN packets from the outside using the **state** option with NEW. Drop any new connections on the **eth0** connection (assumes only **eth0** is connected to the Internet or outside network).

```
# prevent outside initiated connections
iptables -A INPUT -m state --state NEW -i eth0 -j DROP
iptables -A FORWARD -m state --state NEW -i eth0 -j DROP
```

To allow interaction by the internal network with the firewall, you allow input by all packets on the internal Ethernet connection, **eth1**. The valid internal network addresses are designated as the input source:

```
iptables -A INPUT -j ACCEPT -p all -i eth1 -s 192.168.0.0/24
```

To implement masquerading, where systems on the private network can use the gateway's Internet address to connect to Internet hosts, you create a NAT table (**-t nat**) POSTROUTING rule with a MASQUERADE target:

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

In addition, to allow ping and destination-unreachable ICMP packets, you enter INPUT rules with the firewall as the destination. To enable ping operations, you use both echo-reply and echo-request ICMP types, and for destination unreachable, you use the destination-unreachable type:

```
iptables -A INPUT -j ACCEPT  -p icmp -i eth0 --icmp-type \
   echo-reply -d 10.0.0.1
iptables -A INPUT -j ACCEPT  -p icmp -i eth0 --icmp-type \
   echo-request -d 10.0.0.1
iptables -A INPUT -j ACCEPT -p icmp -i eth0 --icmp-type \
   destination-unreachable -d 10.0.0.1
```

At the end, IP forwarding is turned on again:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

A listing of these iptables options shows the different rules for each option, as shown here:

```
# iptables -L
Chain INPUT (policy DROP)
target   prot opt source           destination
LOG      all  -- !192.168.0.0/24  anywhere        LOG level warning
DROP     all  -- !192.168.0.0/24  anywhere
DROP     all  --  192.168.0.0/24  anywhere
DROP     all  --  127.0.0.0/8     anywhere
ACCEPT   all  --  anywhere        anywhere
ACCEPT   tcp  --  10.0.0.2        anywhere        tcp dpt:http
ACCEPT   tcp  --  10.0.0.2        192.168.0.0/24  state RELATED,ESTABLISHED
                                                             tcp spt:http
ACCEPT  !icmp --  anywhere        anywhere        state RELATED,ESTABLISHED
DROP     all  --  anywhere        anywhere        state NEW
ACCEPT   all  --  192.168.0.0/24  anywhere
ACCEPT   icmp --  anywhere        10.0.0.1        icmp echo-reply
ACCEPT   icmp --  anywhere        10.0.0.1        icmp echo-request
ACCEPT   icmp --  anywhere        10.0.0.1        icmp destination-
unreachable

Chain FORWARD (policy ACCEPT)
target   prot opt source           destination
DROP     all  -- !192.168.0.0/24  anywhere
DROP     all  --  192.168.0.0/24  anywhere
DROP     all  --  127.0.0.0/8     anywhere
DROP     all  --  anywhere        anywhere        state NEW

Chain OUTPUT (policy ACCEPT)
target   prot opt source        destination
DROP       tcp  --  anywhere     192.168.0.0/24  state NEW tcp spt:http
# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target      prot opt source       destination

Chain POSTROUTING (policy ACCEPT)
target      prot opt source       destination
MASQUERADE  all  --  anywhere      anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source       destination
```

For more complex rules, you may want to create your own chain to reduce repetition. A common method is to define a user chain for both INPUT and FORWARD chains, so that you do not have to repeat DROP operations for each. Instead you would have only one user chain that both FORWARD and INPUT chains would feed into for DROP operations. Keep in mind that both FORWARD and INPUT operations may have separate rules in addition to the ones they share. In the next example, the myfilter script has been rewritten, as myfilter2, with a user-defined chain called arriving. The chain is defined with the **–N** option at the top of the script:

```
iptables –N arriving
```

A user chain has to be defined before it can be used as a target in other rules. So, you have to first define and add all the rules for that chain, and then use it as a target. In the myfilter2 script, the arriving chain is defined and its rules added. Then, at the end of the file, it is used as a target for both the INPUT and FORWARD chains. The INPUT chain lists rules for accepting packets, whereas the FORWARD chain has an ACCEPT policy, which will accept them by default.

myfilter2

```
# )# Firewall Gateway system IP address is 10.0.0.1 using Ethernet device
eth0
# Private network address is 192.168.0.0 using Ethernet device eth1
# Web site address is 10.0.0.2

# modprobe iptable_filter
# turn off IP forwarding
echo 0 > /proc/sys/net/ipv4/ip_forward

iptables -N arriving

# Flush chain rules
iptables -F INPUT
iptables -F OUTPUT
iptables -F FORWARD
iptables -F arriving

# set default (policy) rules
iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT

# IP spoofing, deny any packets on the internal network that have an
external
source address.
iptables -A arriving -j LOG  -i eth1 \! -s 192.168.0.0/24
iptables -A arriving -j DROP  -i eth1 \! -s 192.168.0.0/24
# IP spoofing, deny any outside packets (any not on eth1) that have the
source
address of the internal network
iptables -A arriving -j DROP \! -i eth1 -s 192.168.0.0/24
# IP spoofing, deny any outside packets with localhost address
# (packets not on the lo interface (any on eth0 or eth1) that have the
source
address of localhost)
iptables -A arriving -j DROP  -i \! lo  -s  127.0.0.0/255.0.0.0

# allow all incoming messages for users on your firewall system
iptables -A arriving -j ACCEPT  -i lo

# allow  communication to the Web server (address 10.0.0.2), port www
iptables -A arriving  -j ACCEPT -p tcp -i eth0  —dport www -s 10.0.0.2
# Allow  established connections from Web servers to internal network
iptables -A arriving -m state —state ESTABLISHED,RELATED -i eth0 -p tcp  —
sport
www -s 10.0.0.2 -d 192.168.0.0/24  -j ACCEPT
# Prevent new  connections from Web servers to internal network
iptables -A OUTPUT -m state —state  NEW -o eth0 -p tcp  —sport www -d
192.168.0.0/24  -j DROP

# allow established and related outside communication to your system
# allow outside communication to the firewall, except for ICMP packets
iptables -A arriving -m state —state ESTABLISHED,RELATED -i eth0 -p \! icmp
-j
ACCEPT
# prevent outside initiated connections
iptables -A arriving -m state —state NEW -i eth0 -j DROP
```

```
# allow all local communication to and from the firewall on eth1 from the
local
network
iptables -A arriving -j ACCEPT -p all -i eth1 -s 192.168.0.0/24

# Set up masquerading to allow internal machines access to outside network
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

# Accept ICMP Ping (0 and 8) and Destination unreachable (3)  messages
# Others will be rejected by INPUT and OUTPUT DROP policy
iptables -A arriving -j ACCEPT  -p icmp -i eth0 —icmp-type  echo-reply -d
10.0.0.1
iptables -A arriving -j ACCEPT  -p icmp -i eth0 —icmp-type  echo-request -d
10.0.0.1
iptables -A arriving -j ACCEPT -p icmp -i eth0 —icmp-type
destination-unreachable -d 10.0.0.1

iptables -A INPUT -j arriving
iptables -A FORWARD -j arriving

# Turn on IP Forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
```

A listing of the rules is shown here:

```
# iptables -L -n
Chain INPUT (policy DROP)
target    prot opt source          destination
arriving  all  --  0.0.0.0/0         0.0.0.0/0

Chain FORWARD (policy ACCEPT)
target    prot opt source          destination
arriving  all  --  0.0.0.0/0         0.0.0.0/0

Chain OUTPUT (policy ACCEPT)
target    prot opt source          destination
DROP      tcp  --  0.0.0.0/0        192.168.0.0/24  state NEW tcp spt:80

Chain arriving (2 references)
target    prot opt source          destination
LOG       all  -- !192.168.0.0/24   0.0.0.0/0       LOG flags 0 level 4
DROP      all  -- !192.168.0.0/24   0.0.0.0/0
DROP      all  --  192.168.0.0/24   0.0.0.0/0
DROP      all  --  127.0.0.0/8      0.0.0.0/0
ACCEPT    all  --  0.0.0.0/0        0.0.0.0/0
ACCEPT    tcp  --  10.0.0.2         0.0.0.0/0       tcp dpt:80
ACCEPT    tcp  --  10.0.0.2         192.168.0.0/24  state
RELATED,ESTABLISHED
                                                            tcp spt:80
ACCEPT   !icmp --  0.0.0.0/0        0.0.0.0/0       state
RELATED,ESTABLISHED
DROP      all  --  0.0.0.0/0        0.0.0.0/0       state NEW
ACCEPT    all  --  192.168.0.0/24   0.0.0.0/0
ACCEPT    icmp --  0.0.0.0/0        10.0.0.1        icmp type 0
ACCEPT    icmp --  0.0.0.0/0        10.0.0.1        icmp type 8
ACCEPT    icmp --  0.0.0.0/0        10.0.0.1        icmp type 3
```

For rules where chains may differ, you will still need to enter separate rules. In the myfilter2 script, the FORWARD chain has an ACCEPT policy, allowing all forwarded packets to the local network to pass through the firewall. If the FORWARD chain had a DROP policy, like the INPUT chain, then you may need to define separate rules under which the FORWARD chain could accept packets. In this example, the FORWARD and INPUT chains have different rules for accepting packets on the eth1 device. The INPUT rule is more restrictive. To enabled the local network to receive forwarded packets through the firewall, you could enable forwarding on its device using a separate FORWARD rule, as shown here.

```
iptables -A FORWARD -j ACCEPT -p all -i eth1
```

The INPUT chain would accept packets only from those in the local network.

```
iptables -A INPUT -j ACCEPT -p all -i eth1 -s 192.168.0.0/24
```

IP Masquerading

On Linux systems, you can set up a network in which you can have one connection to the Internet, which several systems on your network can use. This way, using only one IP address, several different systems can connect to the Internet. This method is called *IP masquerading,* where a system masquerades as another system, using that system's IP address. In such a network, one system is connected to the Internet with its own IP address, while the other systems are connected on a local area network (LAN) to this system. When a local system wants to access the network, it masquerades as the Internet-connected system, borrowing its IP address.

IP masquerading is implemented on Linux using the ipchains firewalling tool. In effect, you set up a firewall, which you then configure to do IP masquerading. Currently, IP masquerading—as does ipchains firewalling—supports all the common network services, such as Web browsing, telnet, ping, and gopher. Other services, such as IRC, FTP, and Real Audio, require the use of certain modules. Any services you want local systems to access must also be on the firewall system because request and response actually are handled by services on that system.

You can find out more information on IP masquerading at the IP Masquerade Resource Web site at **http://ipmasq.cjb.net/**. In particular, the Linux IP Masquerade mini-HOWTO provides a detailed, step-by-step guide to setting up IP masquerading on your system. IP masquerading must be supported by the kernel before you can use it. If your kernel does not support it, you may have to rebuild the kernel, including IP masquerade support, or use loadable modules to add it. See the IP Masquerade mini-HOWTO for more information.

With IP masquerading, as implemented on Linux systems, the machine with the Internet address is also the firewall and gateway for the LAN of machines that use the firewall's Internet address to connect to the Internet. Firewalls that also implement IP masquerading are sometimes referred to as *MASQ gates.* With IP masquerading, the Internet-connected system (the firewall) listens for Internet requests from hosts on its LAN. When it receives one, it replaces the requesting local host's IP address with the Internet IP address of the firewall and then passes the request out to the Internet, as if the request were its own. Replies from the Internet are then sent to the firewall system. The replies the firewall receives are addressed to the firewall using its Internet address. The firewall then determines the local system to whose request the reply is responding. It then strips off its IP address and sends the response on to the local host across the LAN. The connection is transparent from the perspective of the local machines. They appear to be connected directly to the Internet.

IP masquerading is often used to allow machines on a private network to access the Internet. These could be machines in a home network or a small LAN—say, for a small business. Such a network might have only one machine with Internet access, and as such, only the one Internet address. The local private network would have IP addresses chosen from the private network allocations (10., 172.16., or 192.168.). Ideally, the firewall has two Ethernet cards: one for an interface to the LAN (say, **eth1**) and one for an interface to the Internet, such as **eth0** (for dial-up ISPs, this would be **ppp0** for the modem). The card for the Internet connection (**eth0**) would be assigned the Internet IP address. The Ethernet interface for the local network (**eth1**, in this example) is the firewall Ethernet interface. Your private LAN would have a network address like 192.168.0. Its Ethernet firewall interface (**eth1**) would be assigned the IP address 192.168.0.1. In effect, the firewall interface lets the firewall operate as the local network's gateway. The firewall is then configured to masquerade any packets coming from the private network. Your LAN needs to have its own domain name server, identifying the machines on your network, including your firewall. Each local machine needs to have the firewall specified as its gateway. Try not to use IP aliasing to assign both the firewall and Internet IP addresses to the same physical interface. Use separate interfaces for them, such as two Ethernet cards, or an Ethernet card and a modem (**ppp0**).

Certain services like FTP and IRC can conflict with the IP masquerading setup where your firewall denies new connections. FTP operations use two ports, one to set up the connection and one to handle the data transfer. Connecting on the second port can appear to an IP masqueraded firewall as a new connection attempt, and it will deny it. To overcome this problem, special Netfilter modules are used for specific services. For the FTP service you use ip_masq_ftp and for IRC you use ip_masq_irc. There are also modules for quake, Real Audio (raudio), and VDO live (vdolive).

IP Masquerading with Netfilter (NAT and iptables)

In Netfilter, IP masquerading is a NAT operation and is no longer integrated with packet filtering as in ipchains. IP masquerading commands are placed on the NAT table and treated separately from the packet-filtering commands. To implement IP masquerading with Netfilter, first make sure that the iptable_nat module is loaded (you can have this operation built into the kernel). Normally it is loaded by default.

```
modprobe iptable_nat
```

Then, use iptables to place a masquerade rule on the NAT table. First reference the NAT table with the **-t nat** option. Then add a rule to the POSTROUTING chain with the **-o** option specifying the output device and the **-j** option with the MASQUERADE command:

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Then turn on IP forwarding as you normally would:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Instead of masquerading all local hosts as the single IP address of the firewall/ gateway host, you could use the NAT table to rewrite addresses for a few selected hosts. Such an approach is often applied to setups where you want several local hosts to appear as Internet servers. Using the DNAT and SNAT targets you can direct packets to specific local hosts. You would use rules on the PREROUTING and POSTROUTING chains to direct input and output packets.

For example, the Web server described in the previous example could have been configured as a local host to which a DNAT target could redirect any packets originally received for 10.0.0.2. Say the Web servers were set up on 192.168.0.5. It could appear as having the address 10.0.0.2 on the Internet. Packets sent to 10.0.0.2 would be rewritten and directed to 192.168.0.5 by the NAT table. You would use the PREROUTING chain with the **–d** option to handle incoming packets and POSTROUTING with the **–s** option for outgoing packets.

```
iptables -t nat -A PREROUTING -d 10.0.0.2  \
           --to-destination 192.168.0.5 -j DNAT
iptables -t nat -A POSTROUTING –s 192.168.0.5 \
           --to-source 10.0.0.2 -j SNAT
```

Tip Bear in mind that with iptables, masquerading is no longer combined with the FORWARD chain, as it is with ipchains. So, if you specify a DROP policy for the FORWARD chain, you will also have to specifically enable FORWARD operation for the network that is being masqueraded. You will need both a POSTROUTING rule and FORWARD rule.

IP Masquerading with ipchains

You need to specify forwarding rules for use by ipchains to implement IP masquerading. (See the **mychains** file in the previous section for another example of an ipchains masquerade entry.) The following example assumes the Internet connect host, the firewall, uses its first Ethernet device to connect to the Internet, **eth0**. If you are using a modem to dial up a connection to an ISP, the interface used would probably be the first PPP interface, **ppp0**. The second command appends (**-A**) the forward rule to the target (**-j**) MASQ (masquerade) for the interface (**-i**) **eth0**. The host machines on the LAN must specify the connected system as their gateway machine. The last command enables IP forwarding. To enable IP masquerading using Linuxconf's firewalling entries, select Forward Firewalling and click the Do masquerade check box for any firewall forwarding rules you add that you want to apply to IP masquerading.

```
ipchains -P forward DENY
 ipchains -A forward -i eth0 -j MASQ
 echo 1 > /proc/sys/net/ipv4/ip_forward
```

IP Chains (Kernel 2.2)

IP Chains is the precursor to iptables that was used on Linux systems running the 2.2 kernel. It is still in use on many Linux systems. The Linux Web site for ipchains, which is the successor to ipfwadm used on older versions of Linux, is currently **netfilter. sam**ba.org/ipchains/. Support for ipchains is already implemented on the Linux 2.2 kernel. For earlier versions, you must enable support in the kernel, rebuilding it with network firewalls and IP firewalling features (see Chapter 34). The IP Chains HOWTO, located in your **/usr/doc/ipchains** directory, provides an excellent introduction and tutorial for ipchains and how you use it to implement a firewall. The HOWTO is in Web page format and can be viewed with any Web browser. The HOWTO features specific examples on how to guard against several standard attacks, such as IP spoofing, the ping of death, and teardrop.

Like iptables, ipchains organizes its rules into chains. A *chain* is simply a checklist of rules. These rules specify what action to take for packets containing certain headers. If the packet does match a rule, it is passed to its target, which determines what to do with the packet. If a packet does not match any of the rules, it is passed to the chain's default target. The standard targets are listed in Table 40-7. Unlike iptables, ipchains has both a MASQ and DENY target. A major difference between iptables and ipchains is that ipchains does not treat masquerading as a separate processes. It is just another ipchain rule, not a separate process as in iptables.

A *target* could, in turn, be another chain of rules, even a chain of user-defined rules. A packet could be passed through several chains before finally reaching a target. In the case of user-defined chains, the default target is always the next rule in the chains from which it was called. This sets up a procedure or function call—like flow of control found in programming languages. When a rule has a user-defined chain as its target, then, when activated, that user-defined chain is executed. If no rules are matched, execution returns to the next rule in the originating chain.

Table 40-7: ipchains Targets

Target	**Function**
ACCEPT	Allow packet to pass through the firewall.
DENY	Deny access by the packet (changed to DROP in iptables).
REJECT	Deny access and notify the sender.
MASQ	Masquerade the packet. Used only in the forward chain or chains called from the forward chain. Replaces sender's address with firewall host address (changed to MASQUERADE in iptables and to a NAT task).
REDIRECT	Redirect the packet to a local socket or process on the firewall. Used only in the input chain or chains called from the forward chain (changed to a NAT task in iptables).
RETURN	Jump to the end of the chain and let the default target process it.

The kernel uses three firewall chains: input, output, and forward. Unlike iptables, in ipchains these chains are written in lowercase. When a packet is received through an interface, the input chain is used to determine what to do with it. The kernel then uses its routing information to decide where to send it. If the kernel sends the packet to another host, the forward chain is checked. Before the packet is actually sent, the output chain is also checked.

You add and modify chain rules using the **ipchains** commands. An **ipchains** command consists of the keyword **ipchains**, followed by an argument denoting the command to execute. For example, **ipchains -A** is the command to add a new rule, whereas **ipchains -D** is the command to delete a rule. The **ipchains** options are listed in Table 40-8. Except for the masquerading options, they are much the same as those used in iptables. The following command simply lists the chains along with their rules currently defined for your system. The output shows the default values created by ipchains commands.

```
ipchains -L
Chain input (policy ACCEPT):
Chain forward (policy ACCEPT):
Chain output (policy ACCEPT):
```

Table 40-8: ipchains Options

Option	Function
-A *chain*	Append a rule to a chain.
-D *chain*	Delete matching rule from a chain.
-D *chain rulenum*	Delete rule *rulenum* (1 = first) from *chain*.
-I *chain* **[***rulenum***]**	Insert in *chain* as *rulenum* (default 1 = first).
-R *chain rulenum*	Replace rule *rulenum* (1 = first) in *chain*.
-L [*chain***]**	List the rules in *chain* or all chains.
-F [*chain***]**	Delete (flush) all rules in *chain* or all chains.
-Z [*chain***]**	Zero counters in *chain* or all chains.
-C *chain*	Test this packet on *chain*.
-N *chain*	Create a new user-defined chain.
-X *chain*	Delete a user-defined chain.
-P *chain target*	Change policy on *chain* to *target*.
-M –L	List current masquerading connections (not valid for iptables).
-M -S *tcp tcpfin udp*	Set masquerading timeout values (not valid for iptables).
-h	Display list of commands.
--**v***ersion*	Display version.

Following the command, list the chain to which the rule applies, such as the input, output, or forward chain, or else a user-defined chain. Next, list different options that specify the actions you want taken. Options exist to specify the address a rule is to match on (**-s, -d**) and to specify the target the rule is to execute (**-j**). The **-s** option specifies the source address attached to the packet, while the **-d** option specifies the destination address attached to the packet (where it is coming from and where it is going to). The **-j** option, which stands for *jump,* is used to specify the target to jump to and execute. This is the target that is executed if the packet in a chain matches all the options. The standard targets used for a firewall are ACCEPT and DENY. The ACCEPT target will allow a packet to pass, whereas a DENY target will refuse access.

In the following example, the user adds a rule to the input chain to accept all packets originating from the address 192.168.0.55. Any packets that are received (input) whose source address (**-s**) matches 192.168.0.55 are accepted and passed through as specified by the ACCEPT target, **-j ACCEPT**.

```
ipchains -A input -s 192.168.0.55 -j ACCEPT
```

To deny access from a particular site, simply specify the DENY target (in iptables, this was changed to DROP). Alternatively, you can send a packet back to where it came from using the

REJECT target instead of the DENY target. In the following example, any packet received from **www.myjunk.com** is rejected:

```
ipchains -A input -s www.myjunk.com -j DENY
```

You can specify an individual address using its domain name or its IP number. For a range of addresses, you can use the IP number of their network and the network IP mask. The IP mask can be an IP number or simply the number of bits making up the mask. For example, all of the addresses in network 192.168.0 can be represented by 192.168.0.0/225.255.255.0 or by 192.168.0.0/24. To specify any address, you can use 0.0.0.0/0.0.0.0 or simply 0/0. By default, rules reference any address if no **-s** or **-d** specification exists. The following example accepts messages coming in that are from (source) any host in the 192.168.0.0 network and that are going (destination) anywhere at all (the **-d** option is left out or could be written as **-d 0/0**):

```
ipchains -A input -s 192.168.0.0/24  -j ACCEPT
```

If your system is hosting an Internet service, such as a Web or FTP server, you can use ipchains to control access to it. You can specify a particular service by specifying the port it uses. For a Web server, the port would be www. The names of services and the ports they use are listed in the **/etc/services** file, which maps ports to particular services. For a domain name server, the port would be domain. You can also use the port number if you want, preceding the number with a colon. The following example accepts all messages to the Web server located at 192.168.0.43:

```
ipchains -A input -d 192.168.0.43 www -j ACCEPT
```

With the **!** operator, you can change the effect of a rule to its inverse. The **!** operator works like a "not" logical operator in programming languages. Placing a **!** operator before a **-s** entry matches on any address that is not the specified address: **! -s 192.168.0.66** matches on any packet whose address is not 192.168.0.66. The operator is helpful if you want to restrict by only a few selected sites. The following example restricts access to 192.168.0.66, denying access to all others:

```
ipchains -A input ! -s 192.168.0.66 -j DENY
```

The inverse can apply also to ports, protocols, and devices. For example, to allow access to any port except the Ethernet device **eth2**, you would use

```
ipchains -A input ! eth2 -j ACCEPT
```

The following example denies access to any port except the Web server:

```
ipchains -A input ! www -j DENY
```

You can enter ipchains commands from the shell command line. When you shut down your system, however, these commands will be lost. To save your commands, you can use the ipchains-save script to save them to a file. The recommended file to use is **/etc/ipchains.rules**. The **ipchains-save** command outputs rules to the standard output. To save them in a file, you must redirect the output to a file with the redirection operator, >.

Secure Shell (SSH)

Although a firewall can protect a network from attempts to break in to it from the outside, the problem of securing legitimate communications to the network from outside sources still exists. A particular problem is one of users who want to connect to your network remotely. Such connections could be monitored, and information such as passwords and user IDs used when the user logs into your network could be copied and used later to break in. One solution is to use a Secure Shell (SSH) tool like slogin for remote logins. SSH encrypts any communications between the remote user and a system on your network. The SSH programs are meant to replace remote tools such as rsh and rcp (see Chapter 21), which perform no encryption and introduce security risks such as transmitting passwords in clear text. You can also use SSH to encode X server sessions as well as FTP transmissions (sftp).

Two different implementations of SSH currently use what are, in effect, two different and incompatible protocols. The first version of SSH, known as SSH1, uses the original SSH protocol, whereas version 2.0, known as SSH2, uses a completely rewritten version of the SSH protocol. Encryption is performed in different ways, encrypting different parts of a packet. SSH1 uses server and host keys to authenticate systems, whereas SSH2 only uses host keys. Furthermore, certain functions, such as sftp, are only supported by SSH2.

SSH secures connections by both authenticating users and encrypting their transmissions. The authentication process is handled with public key encryption described in Chapter 6. Once authenticated, transmissions are encrypted by a cipher agreed upon by the SSH server and client for use in a particular session. Authentication is applied to both hosts and users. SSH first authenticates a particular host, verifying that it is a valid SSH host that can be securely communicated with. Then the user is authenticated, verifying that the user is who they say they are.

The public key encryption used in SSH authentication makes use of two keys: a public key and a private key. The *public key* is used to encrypt data, while the *private key* decrypts it. Each host or user has his or her own public and private keys. The public key is distributed to other hosts, who can then use it to encrypt data that only the host's private key can decrypt. For example, when a host sends data to a user on another system, the host encrypts the data with a public key, which it previously received from that user. The data can only be decrypted by the user's corresponding private key. The public key can safely be sent in the open from one host to another, allowing it to be installed safely on different hosts. You think of the process as taking place between a client and a server. When the client sends data to the server, it first encrypts the data using the server's public key. The server can then decrypt the data using its own private key.

The mechanics of authentication in SSH version 1 and version 2 differ slightly. However, the procedure on the part of users is the same. Essentially, a user creates both public and private keys. For this you use the **ssh-keygen** command. The user's public key then has to be distributed to those users that the original user wants access to. Often this is an account a user has on another host. A passphrase further protects access. The original user will need to know the other user's passphrase to access it.

SSH version 1 uses RSA authentication. When a remote user tries to log in to an account, that account is checked to see if it has the remote user's public key. That public key is then used to encrypt a challenge (usually a random number) that can only be decrypted by the remote user's private key. When the remote user receives the encrypted challenge, that user decrypts the challenge with its private key. SSH version 2 can use either RSA or DSA authentication. The remote user will first encrypt a session identifier using its private key, signing it. The encrypted session identifier is then decrypted by the account using the remote user's public key. The session identifier has been previously set up by SSH for that session.

SSH authentication is first carried out with the host and then with users. Each host has its own host, public, and private keys used for authentication. Once the host is authenticated then the user is queried. Each user has their own public and private keys. Users on an SSH server who want to receive connections from remote users will have to keep a list of those remote users' public keys. Similarly, an SSH host will maintain a list of public keys for other SSH hosts.

SSH uses strong encryption methods for which export from the United States may be restricted. Currently, SSH can deal with the following kinds of attacks:

- IP spoofing, where a remote host sends out packets that pretend to come from another, trusted host
- IP source routing, where a host can pretend an IP packet comes from another, trusted host
- DNS spoofing, where an attacker forges name server records
- Interception of clear text passwords and other data by intermediate hosts
- Manipulation of data by people in control of intermediate hosts
- Attacks based on listening to *X* authentication data and spoofed connections to the X11 server

> Note A commercial version of SSH is available from SSH Communications Security, whose Web site is **www.ssh.com**. SSH Communications Security provides SSH free for noncommercial use and sells SSH for commercial use through Datafellows.

The SSH protocol has become an official Internet Engineering Task Force (IETF) standard. A free and open source version is developed and maintained by the OpenSSH project, currently supported by the OpenBSD project. OpenSSH is the version supplied with Red Hat. You can find out more about OpenSSH at **www.openssh.org**. From here, you can download the most recent version, though Red Hat will provide current RPM versions.

SSH Applications

A full set of OpenSSH RPM packages are included with Red Hat distributions. These include the general OpenSSH package (openssh), the OpenSSH server (openssh-server), and the OpenSSH clients (openssh-clients). These packages also require OpenSSL (openssl) with installs the cryptographic libraries that SSH uses. You can easily update them from Red Hat FTP sites or by using the Red Hat Network.

The SSH applications are listed in Table 40-9. They include several client programs and the ssh server. The ssh server (**sshd**) provides secure connections to anyone from the outside using the ssh client to connect. With ssh, users can remotely log in and execute commands using encrypted transmissions. In the same way, with scp, users can copy files from one host to another securely. The ssh server can also invoke the sftp-server to provide encrypted FTP transmissions to those using the sftp client. This client, which only works with ssh version

2.0, operates much like ftp, with many of the same commands (see Chapter 19). Several configuration utilities are also included, such as ssh-add, which adds valid hosts to the authentication agent, and ssh-keygen, which generates the keys used for encryption.

On Red Hat you can start, restart, and stop the sshd server with the **service** command:

```
service sshd restart
```

Table 40-9: SSH Applications

Application	**Description**
ssh	ssh client
sshd	ssh server (daemon)
sftp	sftp client. Version 2 only. Use **?** to list sftp commands
sftp-server	sftp server. Version 2 only
scp	scp client
ssh-keygen	Utility for generating keys. **-h** for help
ssh-add	Add identities to the authentication agent
ssh-agent	SSH authentication agent
ssh-askpass	X Window System utility for querying passwords
ssh-askpass-gnome	Gnome utility for querying passwords
ssh-signer	Signs host-based authentication packets. Version 2 only. Must be suid root (performed by installation)
slogin	Remote login (version 1)

SSH was originally designed to replace remote access operations, such as rlogin, rcp, and telnet, as well as FTP (see Chapter 21). The ssh-clients package contains corresponding SSH clients to replace these applications. With slogin or ssh, you can log in from a remote host to execute commands and run applications, much as you can with rlogin and rsh. With scp, you can copy files between the remote host and a network host, just as with rcp. scftp lets you make secure FTP connections.

For version 2.0, names of the actual applications have a 2 suffix to indicate they are version 2.0 programs. Version 1.0 applications have a 1 as their suffix. During installation, however, links are set for each application to use only the name with the suffix. For example, if you have installed version 2.0, there is a link called scp to the scp2 application. You can then use the link to invoked the application. Using scp starts scp2. Table 40-9 specifies only the link names, as these are the same for each version. Remember, though, some applications, such as sftp, are only available with version 2.0.

SSH Setup

Using SSH involves creating your own public and private keys and then distributing your public key to other users you want to access. These can be different users or simply user accounts of your own that you have on remote systems. Often people remotely log in from a local client into an account on a remote server, say from a home computer to a company computer. Your home computer would be your client account and the account on your company computer would be your server account. On your client account, you would need to generate your public and private keys. Then you would have to place a copy of your public key in the server account. You can do this by simply emailing the key file or copying the file from a floppy disk. Once the account on your server has a copy of your client user's public key, you can access the server account from your client account. You will be also prompted for the server account's passphrase. You will have to know this to access that account. Figure 40-3 illustrates the SSH setup that allows a user george to access the account cecelia.

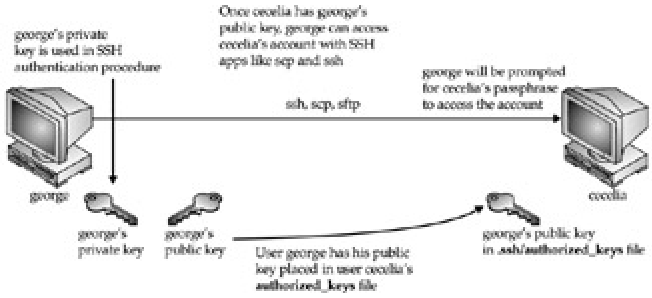

george's private key is used in SSH authentication procedure
Once cecelia has george's public key, george can access cecelia's account with SSH apps like scp and ssh
george will be prompted for cecelia's passphrase to access the account
ssh, scp, sftp
george
cecelia
george's private key
george's public key
User george has his public key placed in user cecelia's authorized_keys file
george's public key in .ssh/authorized_keys file

Figure 40-3: SSH setup and access

The following steps are needed to allow you to use SSH to access other accounts.

1. Create public and private keys on your account along with a passphrase. You will need to use this passphrase to access your account from another account.
2. Distribute your public key to other accounts you want to access, placing them in the **.ssh/authorized_keys** or **.ssh/authorized_keys2** file.
3. Other accounts also have to set up a public and private key along with a passphrase.
4. You will need to also know the other account's passphrase to access it.

You create your public and private keys using the **ssh-keygen** command. The **ssh-keygen** command prompts you for a passphrase, which it will use as a kind of password to protect your private key. The passphrase should be several words long. You are also prompted to enter a filename for the keys. If you do not enter one, SSH will use its defaults. The public key will be given the extension .pub. For SSH version 1, the **ssh-keygen** command generates the public key and places it in your **.ssh/identity.** pub file; it places the private key in the **.ssh/identity** file. For SSH version 2, the **ssh- keygen** command generates the public key and places it in your **.ssh/id_dsa.pub** file; it places the private key in the **.ssh/id_dsa** file.

If you need to change your passphrase, you can do so with the **ssh-keygen** command and the **-p** option. Each user will have his or her own SSH configuration directory, called **.ssh**, located in their own home directory. The public and private keys, as well as SSH configuration files, are placed here. If you build from the source code, then the **make install** operation will automatically run **ssh-keygen**. Table 40-10 lists the SSH configuration files.

Table 40-10: SSH Configuration Files (SSH2 files are placed in /etc/ssh2 and .ssh2 directories)

File	**Description**
$HOME/.ssh/known_hosts	Records host keys for all hosts the user has logged in to (that are not in **/etc/ssh/ssh_known_hosts**).
$HOME/.ssh/random_seed	Used for seeding the random number generator.
$HOME/.ssh/identity	Contains the RSA authentication identity of the user.

Table 40-10: SSH Configuration Files (SSH2 files are placed in /etc/ssh2 and .ssh2 directories)

File	Description
$HOME/.ssh/identity.pub	Contains the public key for authentication (public part of the identity file in human- readable form). The contents of this file should be added to **$HOME/.ssh/authorized_keys** on all machines where you want to log in using RSA authentication.
$HOME/.ssh/config	The per-user configuration file.
$HOME/.ssh/authorized_keys	Lists the RSA keys that can be used for logging in as this user.
/etc/ssh/ssh_known_hosts	Systemwide list of known host keys.
/etc/ssh/ssh_config	Systemwide configuration file. This file provides defaults for those values not specified in the user's configuration file.
/etc/ssh/sshd_config	SSH server configuration file.
$HOME/.rhosts	This file is used in **.rhosts** authentication to list the host/user pairs permitted to log in. Note, this file is also used by rlogin and rsh, which makes using this file insecure.
$HOME/.shosts	This file is used exactly the same way as **.rhosts**. The purpose for having this file is to use rhosts authentication with ssh without permitting login with rlogin or rsh.
/etc/hosts.equiv	This file is used during **.rhosts** authentication. It contains canonical hosts' names, one per line. If the client host is found in this file, login is automatically permitted, provided client and server usernames are the same.
/etc/ssh/shosts.equiv	This file is processed exactly as **/etc/hosts.equiv**. This file may be useful to permit logins using ssh but not using rsh/rlogin.
/etc/ssh/sshrc	System default. Commands in this file are executed by ssh when the user logs in just before the user's shell (or command) is started.
$HOME/.ssh/rc	Commands in this file are executed by ssh when the user logs in just before the user's shell (or command) is started.

A public key is used to authenticate a user and its host. You use the public key on a remote system to allow that user access. In SSH version 2, the public key is placed in the remote user account's **.ssh/authorized_keys2** file. Recall that the public key is held in the **.ssh/id_dsa.pub** file. If a user wants to log in remotely from a local account to an account on a remote system, he or she would first place their public key in the **.ssh/ authorized_keys2** file in the account on the remote system they want to access. If the user **larisa** on **turtle.mytrek.com** wants to access the **aleina** account on **rabbit.mytrek.com, larisa**'s public key from

/home/larisa/.ssh/id_dsa.pub first must be placed in **aleina**'s **authorized_keys2** file, **/home/aleina/.ssh/authorized_keys2**. **larisa** could send the key or have it copied over. A simple **cat** operation can append a key to the authorized key file. In the next example, the user adds the public key for **aleina** in the **larisa.pub** file to the authorized key file. The **larisa.pub** file is a copy of the **/home/larisa/.ssh/id_dsa.pub** file that the user received earlier.

```
cat larisa.pub >>  .ssh/authorized_keys2
```

For SSH version 1, the default name for the authorized key file is simply **.ssh/authorized_keys**.

If you regularly make connections to a variety of remote hosts, you can use the **ssh-agent** command to place private keys in memory where they can be accessed quickly to decrypt received transmissions. The **ssh-agent** command is intended for use at the beginning of a login session. If you are using a shell in your work, use that shell as the argument for the **ssh-agent** command. If you are using the X Window System (Gnome or KDE), use the **startx** command as your argument. That way, any applications you start inherit a connection to **ssh-agent**. For a graphical login, such as GDM, place the **ssh-agent** command in your **.Xclients** file**.** For Gnome, you can use the openssh-askpass-gnome utility, which allows you to enter a password when you log in to Gnome. Gnome will automatically supply that password whenever you use an SSH client. See the Red Hat Customization Guide for details on how to enable openssh-askpass-gnome.

Although the **ssh-agent** command enables you to use private keys in memory, you also must specifically load your private keys into memory using the **ssh-add** command. **ssh-add** with no arguments loads your private key from your **.ssh/identity** file. You are prompted for your passphrase for this private key. To remove the key from memory, use **ssh-add** with the **-d** option. If you have several private keys, you can load them all into memory. **ssh-add** with the **-l** option lists those currently loaded.

SSH also supports the original rhosts form of authentication where hosts and users that are permitted access are placed in an **.rhosts** or **.shosts** file. However, this method is not considered secure.

To use SSH in scripts that run in the background or when you are not logged in, you need to either create a key that has no passphrase or allow automatic access with **.shosts**. To create a key with no passphrase, you run **ssh-keygen** and then press ENTER for the passphrase. Give the key a separate filename. Then, when you invoke SSH use the **–i** option to designate that key. Make sure that the remote user you are accessing has the public key for this key pair. Alternatively you can use **.shosts**. Make sure that the remote user you want to automatically access has your public key. In the remote user's **.shosts** file add an entry for your host and user name.

ssh

With ssh, you can remotely log in from a local client to a remote system on your network operating as the SSH server. The term *local client* here refers to one outside the network such as your home computer, and the term *remote* refers to a host system on the network to which you are connecting. In effect, you connect from your local system to the remote network host. It is designed to replace rlogin, which performs remote logins, and rsh, which executes remote commands. With ssh, you can log in from a local site to a remote host on your network and then send commands to be executed on that host. ssh is also capable of supporting X Window System connections. This feature is automatically enabled if you make an ssh connection from an X Window System environment, such as Gnome or KDE. A connection is set up for you between the local X server and the remote X server. The remote host sets up a dummy X server and sends any X Window System data through it to your local system to be processed by your own local X server.

The ssh login operation function is much like the **rlogin** command. You enter the **ssh** command with the address of the remote host, followed by an **-l** option and the login name (username) of the remote account you are logging into. The following example logs into the **aleina** user account on the **rabbit.mytrek.com** host:

```
ssh rabbit.mytrek.com -l aleina
```

The following listing shows how the user george accesses the cecelia account on **turtle.mytrek.com**.

```
[george@turtle george]$ ssh turtle.mytrek.com -l cecelia
cecelia@turtle.mytrek.com's password:
Last login: Wed Sep 19 15:13:05 2001 from turtle.mytrek.com
[cecelia@turtle cecelia]$
```

A variety of options are available to enable you to configure your connection (see Table 40-11). Most have corresponding configuration options that can be set in the configuration file. For example, with the **-c** option, you can designate which encryption method you want to use. With the **-i** option, you can select a particular private key to use. The **-C** option enables you to have transmissions compressed at specified levels.

Table 40-11: ssh Command Options

Option	**Description**
-a	Disables forwarding of the authentication agent connection.
-c idea|des|3des|blowfish|arcfour|none	Selects the cipher to use for encrypting the session. **idea** is used by default and is believed to be secure. **none** disables encryption entirely; it is only intended for debugging and renders the connection insecure.
-e *ch*|^*ch*|**none**	Sets the escape character for sessions with a pty (default: ~). The escape character is only recognized at the beginning of a line. The escape character followed by a dot (.) closes the connection, followed by CTRL-Z suspends the connection, and followed by itself sends the escape character once. Setting the character to "none" disables any escapes and makes the session fully transparent.
-f	Requests ssh to go to background after authentication is done and forwardings are established.
-i *identity_file*	Selects the file from which the identity (private key) for RSA authentication is read. Default is **.ssh/identity** in the user's home directory.

Table 40-11: ssh Command Options

Option	Description
-k	Disables forwarding of the Kerberos tickets.
-l *login_name*	Specifies the user to log in as on the remote machine.
-n	Redirects stdin from **/dev/null** (actually, prevents reading from stdin). Used when ssh is run in the background. Used to run X11 programs in a remote machine.
-o *option*	Specifies options.
-p *port*	Port to connect to on the remote host.
-q	Quiet mode, suppresses warning and diagnostic messages.
-P	Uses nonprivileged port.
-t	Forces pseudo-tty allocation.
-v	Verbose mode.
-V	Displays version number.
-g	Allows remote hosts to connect local port-forwarding ports.
-x	Disables X11 forwarding.
-C	Requests compression of all data. The compression algorithm is the same used by gzip and the level can be controlled by the CompressionLevel option.
-L *port*:*host*:*hostport*	Specifies that the given port on the local (client) host is to be forwarded to the given host and port on the remote side.
-R *port*:*host*:*hostport*	Specifies that the given port on the remote (server) host is to be forwarded to the given host and port on the local side.

scp

You use scp to copy files from one host to another on a network. Designed to replace rcp, scp actually uses ssh to transfer data and employs the same authentication and encryption (see Table 40-12). If authentication requires it, scp requests a password or passphrase. scp operates much like rcp. Directories and files on remote hosts are specified using the username and the host address before the filename or directory. The username specifies the remote user account that scp is accessing, and the host is the remote system where that account is located. You separate the user from the host address with a @, and you separate the host address from the file or directory name with a colon, **:**. The following example copies the file **party** from a user current directory to the user **aleina**'s **birthday** directory, located on the **rabbit.mytrek.com** host:

```
scp party aleina@rabbit.mytrek.com:/birthday/party
```

Table 40-12: scp Options

Option	Description
-q	Turns off statistics display.
-Q	Turns on statistics display.
-r	Recursively copies entire directories.

Table 40-12: scp Options

Option	Description
-v	Verbose mode.
-B	Selects batch mode (prevents asking for passwords or pass-phrases).
-C	Compression-enabled.
-P *port*	Specifies the port to connect to on the remote host.
-S *path-to-ssh*	Specifies the path to **sch** program.

Of particular interest is the **-r** option (recursive), which enables you to copy whole directories. In the next example, the user copies the entire **reports** directory to the user **justin**'s **projects** directory:

```
scp -r reports justin@rabbit.mytrek.com:/projects
```

In the next example, the user george copies the **mydoc1** file form the user cecelia's home directory.

```
[george@turtle george]$ scp cecelia@turtle.mytrek.com:mydoc1  .
cecelia@turtle.mytrek.com's password:
mydoc1     0% |                             |   0 --:--
ETA
mydoc1   100% |*****************************|  17 00:00
[george@turtle george]$
```

Port Forwarding

If, for some reason, you can only connect to a secure host by going through an unsecure host, ssh provides a feature called port forwarding. With *port forwarding,* you can secure the unsecure segment of your connection. This involves simply specifying the port at which the unsecure host is to connect to the secure one. This sets up a direct connection between the local host and the remote host, through the intermediary unsecure host. Encrypted data is passed through directly.

You can set up port forwarding to a port on the remote system or your local system. To forward a port on the remote system to a port on your local system, use the **ssh -R** option, followed by an argument holding the local port, the remote host address, and the remote port to be forwarded, each separated by a colon. This works by allocating a socket to listen to the port on the remote side. Whenever a connection is made to this port, the connection is forwarded over the secure channel, and a connection is made to a remote port from the local machine. In the following example, port 22 on the local system is connected to port 23 on the **rabbit.mytrek.com** remote system:

```
ssh -R 22:rabbit.mytrek.com:23
```

To forward a port on your local system to a port on a remote system, use the **ssh -L** option, followed by an argument holding the local port, the remote host address, and the remote port to be forwarded, each two arguments separated by a colon. A socket is allocated to listen to port on the local side. Whenever a connection is made to this port, the connection is forwarded over the secure channel and a connection is made to the remote port on the remote machine. In the following example, port 22 on the local system is connected to port 23 on the **rabbit.mytrek.com** remote system:

```
ssh -L 22:rabbit.mytrek.com:23
```

You can use the LocalForward and RemoteForward options in your **.ssh/config** file to set up port forwarding for particular hosts or to specify a default for all hosts you connect to.

SSH Session

An SSH session can be implemented as a pseudoterminal, much like a telnet connection, or it can be transparent, as is the case with X server connections. With a psuedoterminal, the user can control the connection with a set of escape characters, each beginning with a tilde (~). To end the connection, use a "~." escape sequence. A ~CTRL-Z suspends the session. The escape sequence must be entered in a line of its own. The **~?** lists the available escape sequences you can use.

If no pseudoterminal is set up, the session is transparent. Usually, setting the escape character to "none" makes the session transparent. Binary data can safely be transmitted during transparent sessions. A transparent session ends when the shell or command on the remote system ends and all connections have been closed. X Window System connections are automatically set up as transparent connections. SSH sets up a proxy X server on the remote system.

SSH Configuration

The SSH configuration file for each user is in his or her **.ssh/config** file. The **/etc/ssh/sys_config** file is used to set sitewide defaults. In the configuration file, you can set various options, as listed in Table 40-13. The configuration file is designed to specify options for different remote hosts to which you might connect. It is organized into segments where each segment begins with the keyword **HOST**, followed by the IP address of the host. The following lines hold the options you have set for that host. A segment ends at the next **HOST** entry. Of particular interest are the **User** and **Cipher** options. Use the **User** option to specify the names of users on the remote system that are allowed access. With the **Cipher** option, you can select which encryption method to use for a particular host (see Table 40-14). The following example allows access from **larisa** at **turtle.mytrek.com** and uses blowfish encryption for transmissions:

```
Host turtle.mytrek.com
     User larisa
     Compression no
     Cipher blowfish
```

Table 40-13: SSH Configuration Options

Option	Description
Host *hostname*	Specifies the host to which the following options apply. All options apply to this host until the next **Host** entry. You can specify a range of hosts by using the * and **?** pattern-matching wildcard symbols. A host specified as only * matches on all hosts and can be used to specify

Table 40-13: SSH Configuration Options

Option	Description
	global options.
BatchMode (yes | no)	If set to yes, passphrase/password querying is disabled. This option is useful in scripts and other batch jobs where you have no user to supply the password. The argument must be yes or no.
Cipher *cipher*	Specifies the cipher to use for encrypting the session, such as idea, des, 3des, blowfish, arcfour, and none. The default is idea (or 3des if idea is not supported by both hosts). Using "none" (no encryption) renders the connection insecure and is intended only for debugging.
ClearAllForwardings (yes | no)	Clears all forwardings after reading all config files and parsing the command line. This is useful to disable forwardings in config files when you want to make a second connection to the host having forwardings in the config file. scp sets this on by default, so it does not fail, even if you have some forwardings set in a config file.
Compression (yes | no)	Specifies whether to use compression.
CompressionLevel (*1–9*)	Specifies the compression level. The argument must be an integer from 1 (fast) to 9 (slow, best). The default level is 6.
ConnectionAttempts *num*	Specifies the number of tries (one per second) to make.
EscapeChar	Sets the escape character (default: ~). The argument should be a single character, ^, followed by a letter or "none" to disable the escape character entirely (making the connection transparent for binary data).
FallBackToRsh (yes | no)	Specifies if connecting with ssh fails because no sshd is on the remote host, rsh should automatically be used instead.
ForwardAgent (yes | no)	Specifies whether the connection to the authentication agent should be forwarded to the remote machine.
ForwardX11 (yes | no)	Specifies whether X11 connections should be automatically redirected over the secure channel and DISPLAY set.
GatewayPorts (yes | no)	Specifies that remote hosts may connect to locally forwarded ports.
GlobalKnownHostsFile *file*	Specifies a file to use instead of **/etc/ssh/ssh_known_hosts**.
HostName *hostname*	Specifies the real hostname to log into. This can be used to specify nicknames or abbreviations for hosts.
IdentityFile *file*	Specifies the file from which the user's RSA authentication identity is read (default **.ssh/identity** in the user's home directory).

Table 40-13: SSH Configuration Options

Option	Description
KeepAlive (yes | no)	Specifies whether the system should send keep alive messages to the other side. Used to detect if connection fails.
KerberosAuthentication (yes | no)	Specifies whether Kerberos V5 authentication should be used.
KerberosTgtPassing (yes | no)	Specifies whether a Kerberos V5 TGT is to be forwarded to the server.
LocalForward *host host***:***port*	Specifies a TCP/IP port on the local machine be forwarded over the secure channel to given host:port from the remote machine. The first argument must be a port number, and the second must be host:port.
NumberOfPasswordPrompts *num*	Specifies the number of password prompts before giving up.
PasswordAuthentication (yes | no)	Specifies whether to use password authentication.
PasswordPromptHost (yes | no)	Specifies whether to include the remote hostname in the password prompt.
PasswordPrompt Login (yes | no)	Specifies whether to include the remote login name in the password prompt.
Port	Specifies the port number to connect on the remote host. Default is 22.
ProxyCommand *command*	Specifies the command to use to connect to the server.
RemoteForward *host host*:*port*	Specifies a TCP/IP port on the remote machine be forwarded over the secure channel to given host:port from the local machine. The first argument must be a port number, and the second must be host:port.
RhostsAuthentication (yes | no)	Specifies whether to try rhosts-based authentication.
RhostsRSAAuthentication (yes | no)	Specifies whether to try rhosts-based authentication with RSA host authentication.
RSAAuthentication (yes | no)	Specifies whether to try RSA authentication.
StrictHostKeyChecking (yes | no | *ask*)	If this flag is set to yes, ssh never automatically adds host keys to the **$HOME/.ssh/known_hosts** file and refuses to connect hosts whose host key has changed.
TISAuthentication (yes | no)	Specifies whether to try TIS authentication.
UsePrivilegedPort (yes | no)	Specifies whether to use a privileged port when connecting to other end.
User *username*	Specifies the user login name.

Table 40-13: SSH Configuration Options

Option	Description
UserKnownHostsFile *file*	Specifies a file to use instead of **$HOME/.ssh/known_hosts**.
UseRsh (yes | no)	Specifies rlogin/rsh should be used for this host if it does not support the ssh protocol.
XAuthLocation *path*	Specifies the path to xauth program.

Table 40-14: SSH2 Encryption Methods

Method	Description
idea	Believed to be secure.
DES	The data encryption standard, but this is breakable by governments, large corporations, and major criminal organizations.
3des (triple-des)	Encrypt-decrypt-encrypt triple with three different keys. Presumably more secure than DES and used as the default if both sites do not support IDEA.
Blowfish	128-bit key encryption algorithm invented by Bruce Schneier.
Arcfour	Equivalent with the RC4 cipher from RSA Data Security (RC4 is a trademark of RSA Data Security). This is the fastest algorithm currently supported.
Twofish	Version 2 only.
Arcfour	Version 2 only.
cast128-cbc	Version 2 only.

To specify global options that apply to any host you connect to, create a **HOST** entry with the asterisk as its host, **HOST ***. This entry must be placed at the end of the configuration file because an option is changed only the first time it is set. Any subsequent entries for an option are ignored. Because a host matches on both its own entry and the global one, its specific entry should come before the global entry. The asterisk, *****, and the question mark, **?**, are both wildcard matching operators that enable you to specify a group of hosts with the same suffix or prefix.

```
Host *
  FallBackToRsh yes
  KeepAlive no
  Cipher idea
```

Kerberos Authentication and Encryption

Kerberos is a network authentication protocol that provides encrypted authentication to connections between a client and a server. As an authentication protocol, Kerberos requires a client to prove his or her identity using encryption methods before they can access a server. Once authenticated, the client and server can conduct all communications using encryption. Whereas firewalls only protect from outside attacks, Kerberos is designed to also protect from attacks from those inside the network. Users already within a network could try to break in to local servers. Kerberos places protection around the servers themselves, rather than an entire network or computer. A free version is available from the Massachusetts Institute of

Technology at **web.mit.edu/kerberos** under the MIT Public License, which is similar to the GNU Public License. The name *Kerberos* comes from Greek mythology and is the name of the three-headed watchdog for Hades.

The key to Kerberos is a Kerberos server through which all requests for any server services are channeled. The Kerberos server then authenticates a client, identifying the client and validating the client's right to use a particular server. The server maintains a database of authorized users. Kerberos then issues the client an encrypted ticket that the client can use to gain access to the server. For example, if a user needs to check his or her mail, a request for use of the mail server is sent to the Kerberos server, which then authenticates the user and issues a ticket that is then used to access the mail server. Without a Kerberos-issued ticket, no one can access any of the servers. Originally, this process required that the user undergo a separate authentication procedure for each server he or she wanted to access. However, users now only need to perform an initial authentication that is valid for all servers.

This process actually involves the use of two servers, an authentication server (AS) and a ticket granting server (TGS). Together they make up what is known as the key distribution center (KDC). In effect, they distribute keys used to unlock access to services. The authentication server first validates a user's identity. The AS issues a ticket called the ticket granting ticket (TGT) that allows the user to access the ticket granting server. The TGS then issues the user another ticket to actually access a service. This way, the user never has any direct access of any kind to a server during the authentication process. The process is somewhat more complex than described. An authenticator using information such as the current time, a checksum, and an optional encryption key is sent along with the ticket and is decrypted with the session key. This authenticator is used by a service to verify the user's identity.

The authentication server validates a user using information in its user database. Each user needs to be registered in the authentication server's database. The database will include a user password and other user information. To access the authentication server, the user provides the username and the password. The password is used to generate a user key with which communication between the AS and the user is encrypted. The user will have its own copy of the user key with which to decrypt communications. The authentication process is illustrated in Figure 40-4.

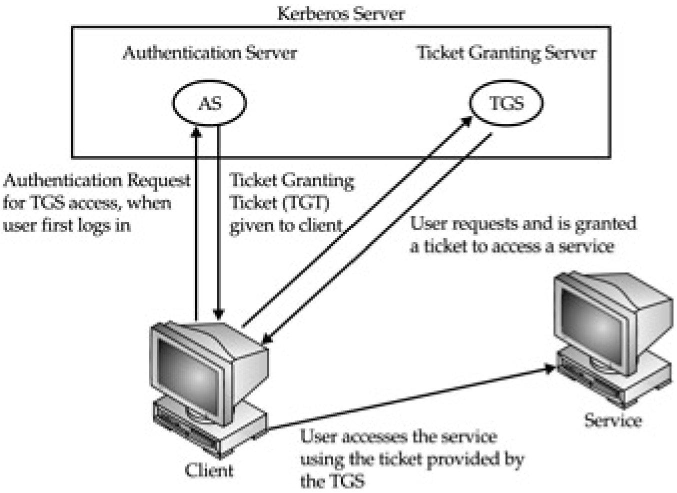

Kerberos Server
Authentication Server
Ticket Granting Server
AS
TGS
Authentication Request for TGS access, when user first logs in
Ticket Granting Ticket (TGT) given to client
User requests and is granted a ticket to access a service
Service
Client
User accesses the service using the ticket provided by the TGS

Figure 40-4: Kerberos authentication

Accessing a service with Kerberos involves the following steps:

1. First the user has to be validated by the authentication server and granted access to the ticket granting server with a ticket access key. The user does this by issuing the **kinit** command, which will ask you enter your Kerberos username and then send it on to the authentication server (the Kerberos username is usually the same as your username).

   ```
   kinit
   ```

2. The AS generates a ticket granting ticket with which to access the ticket granting server. This ticket will include a session key that will be used to let you access the TGS server. The TGT is sent back to you encrypted with your user key (password).
3. The kinit program then prompts you to enter your Kerberos password, which it then uses to decrypt the TGT.
4. Now you can use a client program such as a mail client program to access the mail server, for instance. When you do so, the TGT is used to access the TGS server, which then will generate a ticket for accessing the mail server. The TGS will generate a new session key for use with just the mail server. This will be provided in the ticket sent to you for accessing the mail server. In effect, there is a TGT sessions key used for accessing the TGS server, and a mail session key used for accessing the mail server. The ticket for the mail server is sent to you encrypted with the TGS session key.
5. The client then uses the mail ticket received from the TGS server to access the mail server.
6. If you want to use another service such as FTP, when your FTP client sends a request to the TGS server for a ticket, the TGS server will automatically obtain authorization from the authentication server and issue an FTP ticket with an FTP session key. This kind of support remains in effect for a limited period of time, usually several hours, after which you again have to use kinit to undergo the authentication process and access the TGS server.

Installing and configuring a Kerberos server can be a complex process. Carefully check the documentation for installing the current versions. Some of the key areas are listed here. In the Kerberos configuration file, **krb5.conf**, you can set such features as the encryption method used and the database name. When installing Kerberos, be sure to carefully follow the instructions for providing administrative access. The **/etc/services** file should contain a listing of all Kerberized services. These are services such as kftp or klogin that provide Kerberos FTP and login services. To run Kerberos, you start the Kerberos server with the **krb5kdc** and **kadmin** commands.

Note On Gnome, you can use the krb5 tool to manage Kerberos tickets, and the gkadmin tool to manage Kerberos realms. These are part of the gnome-kerberos package on your Red Hat distribution.

Setting up a particular service to use Kerberos (known as Kerberizing) can be a complicated process. A Kerberized service needs to check the user's identity and credentials, check for a ticket for the service, and, if one is not present, obtain one. Once the service is set up, use of Kerberized services is nearly transparent to the user. Tickets are automatically issued and authentication carried out without any extra effort by the user.

Note The Red Hat Reference Manual provides detailed instructions on setting up Kerberos servers and clients on Red Hat systems.

Web Chapter 41: Shell Programming

Overview

A shell program combines Linux commands in such a way as to perform a specific task. The Linux shell provides you with many programming tools with which to create shell programs. You can define variables and assign values to them. You can also define variables in a script file and have a user interactively enter values for them when the script is executed. There are loop and conditional control structures that repeat Linux commands or make decisions on which commands you want to execute. You can also construct expressions that perform arithmetic or comparison operations. All these programming tools operate like those found in other programming languages.

You can combine shell variables, control structures, expressions, and Linux commands to form a shell program. Usually, the instructions making up a shell program are entered into a script file that can then be executed. You can create this script file using any standard editor. To run the shell program, you then execute its script file. You can even distribute your program among several script files, one of which will contain instructions to execute others. You can think of variables, expressions, and control structures as tools you use to bring together several Linux commands into one operation. In this sense, a shell program is a new, complex Linux command that you have created. The BASH, TCSH, and Z shells that are supported on Linux distributions each have their own shell programming language with slightly different syntax. This chapter discusses BASH shell programming. Table 1 lists several commonly used BASH shell commands discussed throughout this chapter and previously in Chapter 11.

Table 1: BASH: Shell Commands and Arguments

BASH Shell Commands	**Effect**
break	Exits from **for**, **while**, or **until** loop
continue	Skips remaining commands in loop and continues with next iteration
echo	Displays values **-n** eliminates output of new line
eval	Executes the command line
exec	Executes command in place of current process; does not generate a new subshell, but uses the current one
exit	Exits from the current shell
export ***var***	Generates a copy of **var** variable for each new subshell (call-by-value)
history	Lists recent history events
let "*expression***"**	Evaluates an arithmetic, relational, or assignment expression using operators listed in Table 41-3. The expression must be quoted

Table 1: BASH: Shell Commands and Arguments

BASH Shell Commands	**Effect**
read	Reads a line from the standard input
return	Exits from a function
set	Assigns new values for these arguments (when used with command line arguments); lists all defined variables (when used alone)
shift	Moves each command line argument to the left so that the number used to reference it is one less than before; argument $3 would then be referenced by $2, and so on; $1 is lost
test *value option value* **[** *value option value* **]**	Compares two arguments; used as the Linux command tested in control structures **test 2 -eq $count** **[2 -eq $count]**
unset	Undefines a variable
Command Line Arguments	
$0	Name of Linux command
$n	The *n*th command line argument beginning from 1, $1–$n; you can use **set** to change them
$*	All the command line arguments beginning from 1; you can use **set** to change them
$@	The command line arguments individually quoted
$#	The count of the command line arguments
Process Variables	
$$	The PID number, process ID, of the current process
$!	The PID number of the most recent background job
$?	The exit status of the last Linux command executed

Shell Scripts: Commands and Comments

A shell script is a text file that contains Linux commands, which you enter using any standard editor. You can then execute the commands in the file by using the filename as an argument to any **sh** or dot command (**.**). They read the commands in shell scripts and execute them. You can also make the script file itself executable and use its name directly on the command line as you would use the name of any command. To better identify your shell scripts, you can add the **.sh** extension to them, as in **hello.sh**. However, this is not necessary.

You make a script file executable by setting its execute permission using the **chmod** command. The executable permission for the **chmod** command can be set using either symbolic or absolute references. The symbolic reference **u+x** sets the execute permission of a file. The command **chmod u+x hello** will set the execute permission of the **hello** script file. You can now use the script filename **hello** as if it were a Linux command. You only need to set the executable permission once. Once set, it remains set until you explicitly change it. The contents of the **hello** script are shown here.

hello

```
echo "Hello, how are you"
```

The user then sets the execute permission and runs the script using just the script name, in this case, hello.

```
$ chmod u+x hello
$ hello
Hello, how are you
$
```

An absolute reference will set read and write permissions at the same time that it sets the execute permission. See Chapter 12 for a more detailed explanation of absolute and symbolic permission references. In brief, a 700 will set execute as well as read and write permissions for the user; 500 will set only execute and read permissions; 300 only execute and write permissions; and 400 only execute permission. Users most often set 700 or 500. In the next example, the user sets the execute permission using an absolute reference:

```
$ chmod 750 hello
$ hello
Hello, how are you
$
```

It is often helpful to include in a script file short explanations describing what the file's task is as well as describing the purpose of certain commands and variables. You can enter such explanations using comments. A *comment* is any line or part of a line preceded by a sharp (or hash) sign, #, with the exception of the first line. The end of the comment is the next newline character, the end of the line. Any characters entered on a line after a sharp sign will be ignored by the shell. The first line is reserved for identification of the shell, as noted in the following discussion. In the next example, a comment describing the name and function of the script is placed at the head of the file.

hello

```
# The hello script says hello
echo "Hello, how are you"
```

You may want to be able to execute a script that is written for one of the Linux shells while you are working in another. Suppose you are currently in the TCSH shell and want to execute a script you wrote in the BASH shell that contains BASH shell commands. First you would have to change to the BASH shell with the **sh** command, execute the script, and then change back to the TCSH shell. You can, however, automate this process by placing, as the first characters in your script, **#!**, followed by the pathname for the shell program on your system.

Your shell always examines the first character of a script to find out what type of script it is—a BASH, PDKSH, or TCSH shell script. If the first character is a space, the script is assumed to be either a BASH or PDKSH shell script. If there is a # alone, the script is a TCSH shell script. If, however, the # is followed by a **!** character, then your shell reads the pathname of a shell program that follows. A **#!** should always be followed by the pathname of a shell program identifying the type of shell the script works in. If you are currently in a different shell, that shell will read the pathname of the shell program, change to that shell, and execute your script. If you are in a different shell, the space or # alone is not enough to identify a BASH or TCSH shell script. Such identification works only in their own shells. To identify a script from a different shell, you need to include the #! characters followed by a pathname.

For example, if you put **#!/bin/sh** at the beginning of the first line of the **hello** script, you could execute it directly from the TCSH shell. The script will first change to the BASH shell, execute its commands, and then return to the TCSH shell (or whatever type of shell it was executed from). In the next example, the **hello** script includes the **#!/bin/sh** command.

hello

```
#!/bin/sh
# The hello script says hello
echo "Hello, how are you"
```

The user then executes the script while in the TCSH shell.

```
> hello
Hello, how are you
```

Variables and Scripts

In the shell, you can create shell programs using variables and scripts. Within a shell program, you can define variables and assign values to them. Variables are used extensively in script input and output operations. The **read** command allows the user to interactively enter a value for a variable. Often **read** is combined with a prompt notifying the user when to enter a response. Another form of script input, called the Here document, allows you to use lines in a script as input to a command (discussed later). This overcomes the need to always read input from an outside source such as a file.

Definition and Evaluation of Variables: =, $, set, unset

A variable is defined in a shell when you first use the variable's name. A variable name may be any set of alphabetic characters, including the underscore. The name may also include a number, but the number cannot be the first character in the name. A name may not have any other type of character, such as an exclamation point, ampersand, or even a space. Such symbols are reserved by a shell for its own use. A name may not include more than one word, because a shell uses spaces to parse commands, delimiting command names and arguments.

You assign a value to a variable with the assignment operator. You type the variable name, the assignment operator, =, and then the value assigned. Note that you cannot place any spaces around the assignment operator. Any set of characters can be assigned to a variable. In the next example, the **greeting** variable is assigned the string "Hello":

```
$ greeting="Hello"
```

Once you have assigned a value to a variable, you can then use that variable to reference the value. Often, you use the values of variables as arguments for a command. You can reference the value of a variable using the variable name preceded by the **$** operator. The dollar sign is a special operator that uses a variable name to reference a variable's value, in effect evaluating the variable. Evaluation retrieves a variable's value—a set of characters. This set of characters then replaces the variable name on the command line. Thus, wherever a **$** is placed before the variable name, the variable name is replaced with the value of the variable.

In the next example, the shell variable **greeting** is evaluated and its contents, "Hello", are then used as the argument for an **echo** command. The **echo** command simply echoes or prints a set of characters to the screen.

```
$ echo $greeting
Hello
```

Tip You can obtain a list of all the defined variables with the **set** command. If you decide that you do not want a certain variable, you can remove it with the **unset** command.

Variable Values: Strings

The values that you assign to variables may consist of any set of characters. These characters may be a character string that you explicitly type in or the result obtained from executing a Linux command. In most cases, you will need to quote your values using either single quotes, double quotes, backslashes, or back quotes. Single quotes, double quotes, and backslashes allow you to quote strings in different ways. Back quotes have the special function of executing a Linux command and using the results as arguments on the command line.

Although variable values can be made up of any characters, problems occur when you want to include characters that are also used by the shell as operators. Your shell has certain special characters that it uses in evaluating the command line. If you want to use any of these characters as part of the value of a variable, you must first quote them. Quoting a special character on a command line makes it just another character.

- A space is used to parse arguments on the command line.
- The asterisk, question mark, and brackets are special characters used to generate lists of filenames.
- The period represents the current directory.
- The dollar sign is used to evaluate variables, and the greater-than and less-than characters are redirection operators.
- The ampersand is used to execute background commands, and the vertical bar pipes execute output. Double and single quotes allow you to quote several special characters at a time. Any special characters within double or single quotes are quoted. A backslash quotes a single character—the one that it precedes. If you want to assign more than one word to a variable, you need to quote the spaces separating the words. You can do so by enclosing the words within double quotes. You can think of this as creating a character string to be assigned to the variable. Of course, any other special characters enclosed within the double quotes will also be quoted.

The following examples show three ways of quoting strings. In the first example, the double quotes enclose words separated by spaces. Because the spaces are enclosed within double quotes, they are treated as characters—not as delimiters used to parse command line arguments. In the second example, single quotes also enclose a period, treating it as just a character. In the third example, an asterisk is also enclosed within the double quotes. The asterisk is considered just another character in the string and is not evaluated.

```
$ notice="The meeting will be tomorrow"
$ echo $notice
The meeting will be tomorrow

$ message='The project is on time.'
$ echo $message
The project is on time.

$ notice="You can get a list of files with ls *.c"
$ echo $notice
You can get a list of files with ls *.c
```

Double quotes, however, do not quote the dollar sign—the operator that evaluates variables. A **$** next to a variable name enclosed within double quotes will still be evaluated, replacing the variable name with its value. The value of the variable will then become part of the string, not the variable name. There may be times when you want a variable within quotes to be evaluated. In the next example, the double quotes are used so that the winner's name will be included in the notice:

```
$ winner=dylan
$ notice="The person who won is $winner"

$ echo $notice
The person who won is dylan
```

You can always quote any special character, including the **$** operator, by preceding it with a backslash. The backslash is useful when you want to evaluate variables within a string and also include **$** characters. In the next example, the backslash is placed before the dollar sign in order to treat it as a dollar sign character, **\$**. At the same time, the variable **$winner** is evaluated, since double quotes do not themselves quote the **$** operator.

```
$ winner=dylan
$ result="$winner won \$100.00""
$ echo $result
dylan won $100.00
```

Values from Linux Commands: Back Quotes

Though you can create variable values by typing in characters or character strings, you can also obtain values from other Linux commands. However, to assign the result of a Linux command to a variable, you first need to execute the command. If you place a Linux command within back quotes on the command line, that command is executed first and its result becomes an argument on the command line. In the case of assignments, the result of a command can be assigned to a variable by placing the command within back quotes to execute it first.

Tip Think of back quotes as a kind of expression that contains both a command to be executed and its result, which is then assigned to the variable. The characters making up the command itself are not assigned.

In the next example, the command **ls *.c** is executed and its result is then assigned to the variable **listc**. The command **ls *.c** generates a list of all files with a **.c** extension, and this list of files will then be assigned to the **listc** variable.

```
$ listc=`ls *.c`
$ echo $listc
main.c prog.c lib.c
```

Script Input and Output: echo, read, and <<

Within a script, you can use the **echo** command to output data and the **read** command to read input into variables. Also within a script, the **echo** command will send data to the standard output. The data is in the form of a string of characters. As you have seen, the **echo** command can output variable values as well as string constants. The **read** command reads in a value for a variable. It is used to allow a user to interactively input a value for a variable. The **read** command literally reads the next line in the standard input. Everything in the standard input up to the newline character is read in and assigned to a variable. In shell programs, you can combine the **echo** command with the **read** command to prompt the user to enter a value and then read that value into a variable. In the **greetvar** script in the next example, the user is prompted to enter a value for the greeting variable. The **read** command then reads the value the user typed and assigns it to the **greeting** variable.

greetvar

```
echo Please enter a greeting:
read greeting
echo "The greeting you entered was $greeting"
```

The **greetvar** script is then run, as shown here:

```
$ greetvar
Please enter a greeting:
hi
The greeting you entered was hi
$
```

If the value of a variable is a special character and the variable's value is referenced with a **$**, then the special character will be evaluated by the shell. However, placing the evaluated variable within quotes prevents any evaluation of special characters such as **$**. In the **greetvar** script, **$greeting** was placed within a quoted string, preventing evaluation of any special characters. If **$greeting** is not quoted, then any special characters it contains will be evaluated.

The Here operation is a redirection operation, redirecting data within a shell script into a command. It is called Here because the redirected data is here in the shell script, not somewhere else in another file. The Here operation is represented by two less-than signs, <<. The << operator can be thought of as a kind of redirection operator, redirecting lines in a shell script as input to a command. The << operator is placed after the command to which input is being redirected. Lines following the << operator are then taken as input to the command. The end of the input can be specified by an end-of-file character, CTRL-D. Instead of using an end-of- file character, you can specify your own delimiter. A word following the **<<** operator on the same line is taken to be the ending delimiter for the input lines. The delimiter can be any set of symbols. All lines up to the delimiter are read as input to the command.

In the next example, a message is sent to the user mark. The input for the message is obtained from a Here operation. The delimiter for the Here operation is the word **myend**.

mailmark

```
mail mark << myend
Did you remember the meeting
 robert
myend
```

Script Command Line Arguments

Like Linux commands, a shell script can take arguments. When you invoke a script, you can enter arguments on the command line after the script name. These arguments can then be referenced within the script using the **$** operator and the number of its position on the command line. Arguments on the command line are sequentially numbered from 1. The first argument is referenced with **$1**, the second argument with **$2**, and so on. The argument **$0** will contain the name of the shell script, the first word on the command line.

Note These argument references can be thought of as referencing read-only variables. For those familiar with programming terminology, you can think of words on the command line as arguments that are passed into argument variables, **$1** through **$9**.

The argument variables are read-only variables. You cannot assign values to them. Once given the initial values, they cannot be altered. In this sense, argument variables function more as constants—constants determined by the command line arguments. Each word on the command line is parsed into an argument unless it is quoted. If you enter more than one argument, you can reference them with each corresponding argument number. In the next example, four arguments are entered on the command line.

greetargs

```
echo "The first argument is: $1"
echo "The second argument is: $2"
echo "The third argument is: $3"
echo "The fourth argument is: $4"
```

Here is a run of the greetargs script:

```
$ greetargs Hello Hi Salutations "How are you"
The first argument is: Hello
The second argument is: Hi
The third argument is: Salutations
The fourth argument is: How are you
$
```

A set of special arguments allows you to reference different aspects of command line arguments, such as the number of arguments or all the arguments together: **$***, **$@**, **$#**. The **$#** argument contains the number of arguments entered on the command line. This is useful when you need to specify a fixed number of arguments for a script. The argument **$*** references all the arguments in the command line. A command line may have more than nine arguments. The **$@** also references all the arguments on the command line, but allows you to separately quote each one. The difference between **$*** and **$@** is not clear until you use them to reference arguments using the **for-in** control structure. For this reason, they are discussed only briefly here and more extensively in the section on control structures later in the chapter.

Export Variables and Script Shells

When you execute a script file, you initiate a new process that has its own shell. Within this shell you can define variables, execute Linux commands, and even execute other scripts. If you execute another script from within the script currently running, the current script suspends execution, and control is transferred to the other script. All the commands in this other script are first executed before returning to continue with the suspended script. The process of executing one script from another operates much like a function or procedure call in programming languages. You can think of a script calling another script. The calling script waits until the called script finishes execution before continuing with its next command.

Any variable definitions that you place in a script will be defined within the script's shell and only known within that script's shell. Variable definitions are local to their own shells. In a sense, the variable is hidden within its shell. Suppose, however, you want to be able to define a variable within a script and use it in any scripts it may call. You cannot do this directly, but you can export a variable definition from one shell to another using the **export** command. When the **export** command is applied to a variable, it will instruct the system to define a copy of that variable for each new subshell generated. Each new subshell will have its own copy of the exported variable. In the next example, the **myname** variable is defined and exported:

```
$ myname="Charles"
$ export myname
```

Note It is a mistake to think of exported variables as global variables. A shell can never reference a variable outside of itself. Instead, a copy of the variable with its value is generated for the new shell. An exported variable operates to some extent like a scoped global parameter. It is copied to any shell derived from its own shell. Any shell script called directly or indirectly after the exported variable's shell will be given a copy of the exported variable with the initial value.

In the BASH shell, an environment variable can be thought of as a regular variable with added capabilities. To make an environment variable, you apply the **export** command to a variable you have already defined. The **export** command instructs the system to define a copy of that variable for each new subshell generated. Each new subshell will have its own copy of the environment variable. This process is called *exporting variables.*

In the next example, the variable **myfile** is defined in the **dispfile** script. It is then turned into an environment variable using the **export** command. The **myfile** variable will consequently be exported to any subshells, such as that generated when **printfile** is executed.

dispfile

```
myfile="List"
export myfile
echo "Displaying $myfile"
pr -t -n $myfile
printfile
```

printfile

```
echo "Printing $myfile"
lp $myfile &
```

A run of the dispfile script follows:

```
$ dispfile
Displaying List
1 screen
2 modem
3 paper
Printing List
$
```

Arithmetic Shell Operations: let

The **let** command is the BASH shell command for performing operations on arithmetic values. With **let**, you can compare two values or perform arithmetic operations such as addition or multiplication on them. Such operations are used often in shell programs to manage control structures or perform necessary calculations. The **let** command can be indicated either with the keyword **let** or with a set of double parentheses. The syntax consists of the keyword **let** followed by two numeric values separated by an arithmetic or relational operator, as shown here:

```
$ let value1 operator value2
```

You can use as your operator any of those listed in Table 2. The **let** command automatically assumes that operators are arithmetic or relational. You do not have to quote shell-like operators. The **let** command also automatically evaluates any variables and converts their values to arithmetic values. This means that you can write your arithmetic operations as simple arithmetic expressions. In the next example, the **let** command multiplies the values 2 and 7. The result is output to the standard output and displayed.

```
$ let 2*7
14
```

Table 2: BASH: Shell Operators

Arithmetic Operators	**Function**
*	Multiplication
/	Division
+	Addition

Table 2: BASH: Shell Operators

Arithmetic Operators	**Function**
-	Subtraction
%	Modulo—results in the remainder of a division
Relational Operators	
	Greater than
	Less than
=	Greater than or equal to
=	Less than or equal to
=	Equal in **expr**
==	Equal in **let**
!=	Not equal
&	Logical AND
|	Logical OR
!	Logical NOT

If you want to have spaces between operands in the arithmetic expression, you must quote the expression. The **let** command expects one string.

```
$ let "2 * 7"
```

You can also include assignment operations in your **let** expression. In the next example, the result of the multiplication is assigned to **res**:

```
$ let "res = 2 * 7"
$ echo $res
14
$
```

You can also use any of the relational operators to perform comparisons between numeric values, such as checking to see whether one value is less than another. Relational operations are often used to manage control structures such as loops and conditions. In the next example, **helloprg** displays the word "hello" three times. It makes use of a **let** less-than-or-equal operation to manage the loop, **let "again <= 3 "**, and to increment the again variable, **let "again = again + 1"**. Notice that when **again** is incremented, it does not need to be evaluated. No preceding **$** is needed. The **let** command will automatically evaluate variables used in expressions.

helloprg

```
again=1
while let "again <= 3"
do
 echo $again Hello
let "again = again + 1"
done
```

Here is a run of the **helloprg** script.

```
$ helloprg
1 Hello
2 Hello
3 Hello
```

Control Structures

You can control the execution of Linux commands in a shell program with control structures. Control structures allow you to repeat commands and to select certain commands over others. A control structure consists of two major components: a test and commands. If the test is successful, the commands are executed. In this way, you can use control structures to make decisions as to whether commands should be executed.

There are two different kinds of control structures: *loops* and *conditions.* A loop repeats commands, whereas a condition executes a command when certain conditions are met. The BASH shell has three loop control structures: **while**, **for**, and **for-in**. There are two condition structures: **if** and **case**.

The **while** and **if** control structures are more for general purposes, such as performing iterations and making decisions using a variety of different tests. The **case** and **for** control structures are more specialized. The **case** structure is a restricted form of the **if** condition and is often used to implement menus. The **for** structure is a limited type of loop. It runs through a list of values, assigning a new value to a variable with each iteration.

The **if** and **while** control structures have as their test the execution of a Linux command. All Linux commands return an exit status after they have finished executing. If a command is successful, its exit status will be 0. If the command fails for any reason, its exit status will be a positive value referencing the type of failure that occurred. The **if** and **while** control structures check to see if the exit status of a Linux command is 0 or some other value. In the case of the **if** and **while** structures, if the exit status is a zero value, the command was successful and the structure continues.

The test Command

Often you may need to perform a test that compares two values. Yet the test used in control structures is a Linux command, not a relational expression. There is, however, a Linux command called **test** that can perform such a comparison of values. The **test** command will compare two values and return as its exit status a 0 if the comparison is successful.

With the **test** command, you can compare integers, compare strings, and even perform logical operations. The command consists of the keyword **test** followed by the values being compared, separated by an option that specifies what kind of comparison is taking place. The option can be thought of as the operator, but is written, like other options, with a minus sign and letter codes. For example, **-eq** is the option that represents the equality comparison. However, there are two string operations that actually use an operator instead of an option. When you compare two strings for equality you use the equal sign, =. For inequality, you use

!=. Table 3 lists all the options and operators used by **test**. The syntax for the **test** command is shown here:

```
test value -option value
test string = string
```

Table 3: BASH: Shell Test Operators

Integer Comparisons	**Function**
-gt	Greater than
-lt	Less than
-ge	Greater than or equal to
-le	Less than or equal to
-eq	Equal
-ne	Not equal
String Comparisons	
-z	Tests for empty string
-n	Tests for string value
=	Equal strings
!=	Not-equal strings
str	Tests to see if string is not a null string
Logical Operations	
-a	Logical AND
-o	Logical OR
!	Logical NOT
File Tests	
-f	File exists and is a regular file
-s	File is not empty
-r	File is readable
-w	File can be written to, modified
-x	File is executable
-d	Filename is a directory name
-h	Filename is a symbolic link
-c	Filename references a character device
-b	Filename references a block file

In the next example, the user compares two integer values to see if they are equal. In this case, you need to use the equality option, **-eq**. The exit status of the **test** command is examined to find out the result of the test operation. The shell special variable **$?** holds the exit status of the most recently executed Linux command.

```
$ num=5
$ test $num -eq 10
$ echo $?
```

```
1
```

Instead of using the keyword **test** for the **test** command, you can use enclosing brackets. The command **test $greeting = "hi"** can be written as

```
$ [ $greeting = "hi" ]
```

Similarly, the **test** command **test $num -eq 10** can be written as

```
$ [ $num -eq 10 ]
```

The brackets themselves must be surrounded by white space: a space, TAB, or ENTER. Without the spaces, it would be invalid.

The **test** command is used extensively as the Linux command in the test component of control structures. Be sure to keep in mind the different options used for strings and integers. Do not confuse string comparisons and integer comparisons. To compare two strings for equality, you use **=;** to compare two integers, you use the option **-eq**.

Conditions: if, if-else, elif, case

The BASH shell has a set of conditional control structures that allow you to choose what Linux commands to execute. Many of these are similar to conditional control structures found in programming languages, but there are some differences. The **if** condition tests the success of a Linux command, not an expression. Furthermore, the end of an **if-then** command must be indicated with the keyword **fi**, and the end of a **case** command is indicated with the keyword **esac**. The condition control structures are listed in Table 4.

Table 4: BASH: Shell Control Structures

Condition Control Structures: **if, else, elif, case**	**Function**

if *command* **then**
command
fi
	if executes an action if its **test** command is true.

if *command* **then**
command
else
command
fi
	if-else executes an action if the exit status of its **test** command is true; if false, then the **else** action is executed.

if *command* **then**
command
elif *command* **then**
command
else
command
fi
	elif allows you to nest **if** structures, enabling selection among several alternatives; at the first true **if** structure, its commands are executed and control leaves the entire **elif** structure.

case *string* **in** *pattern*) *command;*;	**case** matches the string value to any of several patterns; if a pattern is matched, its associated commands are executed.

Table 4: BASH: Shell Control Structures

Condition Control Structures: **if, else, elif, case**	**Function**
esac	
command **&&** *command*	The logical AND condition returns a true 0 value if both commands return a true 0 value; if one returns a nonzero value, then the AND condition is false and also returns a nonzero value.
command || *command*	The logical OR condition returns a true 0 value if one or the other command returns a true 0 value; if both commands return a nonzero value, then the OR condition is false and also returns a nonzero value.
! *command*	The logical NOT condition inverts the return value of the command.
Loop Control Structures: **while**, **until**, **for**, **for-in**, **select**	

while *command*
do
command
done
	while executes an action as long as its **test** command is true.

until *command*
do
command
done
	until executes an action as long as its **test** command is false.

for *variable* **in** *list-values*
do
command
done
	for-in is designed for use with lists of values; the variable operand is consecutively assigned the values in the list.

for *variable*
do
command
done
	for is designed for reference script arguments; the variable operand is consecutively assigned each argument value.

Loop Control Structures: **while**, **until**, **for**, **for-in**, **select** **(continued)**	

select *string* **in** *item-list*
do
command
done
	select creates a menu based on the items in the *item-list;* then it executes the command; the command is usually a **case**.

if-then

The **if** structure places a condition on commands. That condition is the exit status of a specific Linux command. If a command is successful, returning an exit status of 0, then the commands within the **if** structure are executed. If the exit status is anything other than 0, the command has failed and the commands within the **if** structure are not executed.

The **if** command begins with the keyword **if** and is followed by a Linux command whose exit condition will be evaluated. This command is always executed. After the command, the keyword **then** goes on a line by itself. Any set of commands may then follow. The keyword **fi** ends the command. Often, you need to choose between two alternatives based on whether or not a Linux command is successful. The **else** keyword allows an **if** structure to choose between two alternatives. If the Linux command is successful, those commands following the **then** keyword are executed. If the Linux command fails, those commands following the **else** keyword are executed. The syntax for the **if-then-else** command is shown here:

```
if Linux Command
 then
 Commands
 else
 Commands
fi
```

The **elsels** script in the next example executes the **ls** command to list files with two different possible options, either by size or with all file information. If the user enters an **s**, files are listed by size; otherwise, all file information is listed.

elsels

```
echo Enter s to list file sizes,
echo otherwise all file information is listed.
echo -n "Please enter option: "
read choice
if [ "$choice" = s ]
  then
     ls -s
  else
     ls -l
fi
echo Good-bye
```

A run of the **elsels** script is shown here:

```
$ elsels
Enter s to list file sizes,
otherwise all file information is listed.
Please enter option: s
total 2
 1 monday 2 today
$
```

The **elif** structure allows you to nest **if-then-else** operations. The **elif** structure stands for "else if." With **elif**, you can choose between several alternatives. The first alternative is specified with the **if** structure, followed by other alternatives, each specified by its own **elif** structure. The alternative to the last **elif** structure is specified with an **else**. If the test for the first **if** structure fails, control will be passed to the next **elif** structure, and its test will be executed. If it fails, control is passed to the next **elif** and its test checked. This continues until a test is true. Then that **elif** has its commands executed and control passes out of the **if** structure to the next command after the **fi** keyword.

The Logical Commands: && and ||

The logical commands perform logical operations on two Linux commands. The syntax is as follows:

```
command && command
command || command
```

In the case of the logical AND, **&&**, if both commands are successful, the logical command is successful. For the logical OR, **||**, if either command is successful, the OR is successful and returns an exit status of 0. The logical commands allow you to use logical operations as your test command in control structures.

case

The **case** structure chooses among several possible alternatives. The choice is made by comparing a value with several possible patterns. Each possible value is associated with a set of operations. If a match is found, the associated operations are performed. The **case** structure begins with the keyword **case**, an evaluation of a variable, and the keyword **in**. A set of patterns then follows. Each pattern is a regular expression terminated with a closing parenthesis. After the closing parenthesis, commands associated with this pattern are listed, followed by a double semicolon on a separate line, designating the end of those commands. After all the listed patterns, the keyword **esac** ends the **case** command. The syntax looks like this:

```
case string in
   pattern)
         commands
         ;;
   pattern)
         commands
         ;;
   *)
         default commands
         ;;
 esac
```

A pattern can include any shell special characters. The shell special characters are the *****, **[]**, **?**, and **|**. You can specify a default pattern with a single ***** special character. The ***** special character matches on any pattern and so performs as an effective default option. If all other patterns do not match, the ***** will. In this way, the default option is executed if no other options are chosen. The default is optional. You do not have to put it in.

Loops: while, for-in, for

The BASH shell has a set of loop control structures that allow you to repeat Linux commands. They are the **while**, **for-in**, and **for** structures. Like the BASH **if** structure, **while** and **until** test the result of a Linux command. However, the **for** and **for-in** structures do not perform any test. They simply progress through a list of values, assigning each value in turn to a specified variable. Furthermore, the **while** and **until** structures operate like corresponding structures found in programming languages, whereas the **for** and **for-in** structures are very different. The loop control structures are listed in Table 41-4.

while

The **while** loop repeats commands. A **while** loop begins with the keyword **while** and is followed by a Linux command. The keyword **do** follows on the next line. The end of the loop is specified by the keyword **done**. Here is the syntax for the **while** command:

```
while Linux command
 do
 commands
 done
```

The Linux command used in **while** structures is often a **test** command indicated by enclosing brackets. In the **myname** script, in the next example, you are asked to enter a name. The name is then printed out. The loop is controlled by testing the value of the variable again using the bracket form of the **test** command.

myname

```
again=yes
while [ "$again" = yes ]
 do
   echo -n "Please enter a name: "
   read name
   echo "The name you entered is $name"
   echo -n "Do you wish to continue? "
   read again
 done
 echo Good-bye
```

Here is a run of the **myname** script:

```
$ myname
Please enter a name: George
The name you entered is George
Do you wish to continue? yes
Please enter a name: Robert
The name you entered is Robert
Do you wish to continue? no
Good-bye
```

for-in

The **for-in** structure is designed to reference a list of values sequentially. It takes two operands—a variable and a list of values. The values in the list are assigned one by one to the variable in the **for-in** structure. Like the **while** command, the **for-in** structure is a loop. Each time through the loop, the next value in the list is assigned to the variable. When the end of the list is reached, the loop stops. Like the **while** loop, the body of a **for-in** loop begins with the keyword **do** and ends with the keyword **done**. The syntax for the **for-in** loop is shown here:

```
for variable in list of values
 do
 *commands*
 done
```

In the **mylistfor** script, the user simply outputs a list of each item with today's date. The list of items makes up the list of values read by the **for-in** loop. Each item is consecutively assigned to the **grocery** variable.

mylistfor

```
tdate=`date +%D`
for grocery in milk cookies apples cheese
  do
   echo "$grocery
   $tdate"
  done
```

A run of the **mylistfor** script follows:

```
$ mylistfor
milk 10/23/00
cookies 10/23/00
apples 10/23/00
cheese 10/23/00
$
```

The **for-in** loop is handy for managing files. You can use special characters to generate filenames for use as a list of values in the **for-in** loop. For example, the * special character, by itself, generates a list of all files and directories, and ***.c** lists files with the **.c** extension. The special character * placed in the **for-in** loop's value list will generate a list of values consisting of all the filenames in your current directory.

```
for myfiles in *
 do
```

The **cbackup** script makes a backup of each file and places it in a directory called **sourcebak**. Notice the use of the * special character to generate a list of all filenames with a **.c** extension.

cbackup

```
for backfile in *.c
  do
    cp $backfile sourcebak/$backfile
    echo $backfile
  done
```

A run of the **cbackup** script follows:

```
$ cbackup
io.c
lib.c
```

```
main.c
$
```

for

The **for** structure without a specified list of values takes as its list of values the command line arguments. The arguments specified on the command line when the shell file is invoked become a list of values referenced by the **for** command. The variable used in the **for** command is set automatically to each argument value in sequence. The first time through the loop, the variable is set to the value of the first argument. The second time, it is set to the value of the second argument.

The **for** structure without a specified list is equivalent to the list **$@**. **$@** is a special argument variable whose value is the list of command line arguments. In the next example, a list of C program files is entered on the command line when the shell file **cbackuparg** is invoked. In **cbackuparg**, each argument is automatically referenced by a **for** loop, and **backfile** is the variable used in the **for** loop. The first time through the loop, **$backfile** holds the value of the first argument, **$1**. The second time through, it holds the value of the second argument, **$2**.

cbackuparg

```
for backfile
  do
    cp $backfile sourcebak/$backfile
    echo "$backfile "
  done
```

A run of the **cbackuparg** script is shown here:

```
$ cbackuparg main.c lib.c io.c
main.c
lib.c
io.c
```

Web Chapter 42: Compilers, Libraries, and Programming Tools

Overview

An *application* is an executable program created by a programmer using one of several programming languages. Linux provides several utilities with which a programmer can control development of an application. Foremost among these is the gcc utility, which invokes the compiler for the C and C++ programming languages, generating an executable version of a program. Most Linux applications are written in the C or C++ programming language.

Application development often makes extensive use of libraries. You can create your own libraries or choose from specialized libraries. You can use libraries such as the **X Windows** library to program X Window displays, or the **gdbm** library, with which you can have database access to files. Libraries have become more flexible and can now be shared or loaded dynamically.

Other utilities allow you to better manage the development of your applications. The gdb symbolic debuggers help you to locate runtime errors. indent and cproto help you prepare your source code. Autoconf and RPM help you package your software for distribution. Version control systems such as RCS and CVS help you maintain a record of changes as you develop a software application. Table 1 lists the development tools described in this chapter.

Table 1: Programming: Tools

Tool	**Description**
gcc, g++	GNU C and C++ compiler, **www.gnu.org/software/gcc**
gdb	GNU debugger, **www.gnu.org/software/gdb**
RCS	Revision Control System, **www.gnu.org/software/rcs**
CVS	Control Versions System, **www.cvshome.org**
Man	Online manual documentation
Autoconf	Automatic configuration for source code software compiling, **www.gnu.org/software/autoconf**

Getting Information: info

Though there are Man pages for all the compilers and their tools, much more detailed information is available through the GNU info system. These are files located in the **/usr/info** directory that contain detailed descriptions and examples for various GNU tools. They are the equivalent of compact online manuals. There are info documents for the gcc compiler, the C and C++ libraries, and the Autoconf utility. Other applications may have their own local directories with info files such as the **/usr/TeX/info** directory that holds info files for LaTeX. You invoke the main menu of info documents by entering the command **info**.

```
$ info
```

You then use the SPACEBAR to page down the menu. When you find a topic you want, you press the **m** key. This opens up a line at the bottom of the screen where you can type in the name of the menu item. When you press ENTER, that document comes up. Pressing **b** pages you back to the beginning, and **U** puts you up to the previous menu. The command **info info** will bring up a tutorial on how to use info.

The C Compiler: gcc

There is a special relationship between the Unix operating system and the C programming language. The C programming language was developed specifically as a tool for programming the Unix operating system. The code for the Unix operating system is actually written in C. Linux has the same kind of special relationship. Most Linux systems include the GNU version of the C compiler, gcc. The C programming language is a very complex language with many different features. This section briefly describes the basic components of the C programming language and uses them to construct a useful programming example. With an example program, we can then examine the different ways you can compile C programs.

Note Instead of using the command line interface to compile applications, you can use an integrated development environment (IDE) with a Gnome or KDE interface to compile and edit application source files. On Gnome you can use gbuilder, Anjuta, and Titano. On KDE, you can use Gideon, KDevelop, and KDEStudio.

You invoke the GNU C compiler on your Linux system with the **gcc** command. The **gcc** command, in turn, calls four other components. The first is the preprocessor. A C program contains special preprocessor commands that modify the code before it is sent to the compiler. The second component is the compiler itself. The compiler will process the code and generate an assembly code version of the program. The third component is the assembler. The assembler will use the assembly code version of the program to generate an object code version. The fourth component is the linker. The linker uses the program's object code to generate an executable file. The default name of this executable file is **a.out**. Normally, you should give the executable file a name of your own choosing. The **-o** option takes a filename as its argument. This filename will be the name of the executable file instead of the default, **a.out**. A list of gcc options is provided in Table 2. In the next example, the **gcc** command compiles the program **greet.c**. The user names the executable file "greet". The executable file is run by entering it at the Linux prompt as if it were a command.

```
$ gcc greet.c -o greet
$ greet
Hello, how are you
```

Table 2: The: gcc Utility: the C Compiler

Command	**Description**
gcc	The gcc utility creates an executable program using a preprocessor, a compiler, an assembler, and a linker. The preprocessor executes preprocessing commands found in the source code file. Such commands perform simple text substitutions. The compiler compiles a source code file into assembly code. The assembler then compiles assembly code files into object code files. The linker then links object code files into an executable file.gcc takes as possible arguments source code, object code, and assembly code files as well as several options. gcc recognizes a file by its extension: **.c** C source code files **.o** Object code files **.s** Assembly code files **.m** Objective C **.a** Archive library files
g++	Invokes the cc utility work on C++ source code files. g++ takes as possible arguments source code, object code, and assembly code files as well as several options. gcc recognizes a file by its extension: **.C** C++ files **.cpp** C++ files **.cc** C++ source code files **.o** Object code files **.s** Assembly code files
Option	Function
-S	Output only assembly code. Assembly code versions of compiled

Table 2: The: gcc Utility: the C Compiler

Command	**Description**
	files have extension **.s**. The example will generate a file called **greet.s**.
-P	Output result of preprocessor.
-c	Create object code file only. Object code versions of compiled files have the extension **.o**.
-g	Prepare compiled program for use with symbolic debugger.
-o *filename*	Name the executable file *filename*. Default is **a.out**.
-O	Optimize compilation.
-l *filename*	Link system library by name of filename. The filename is preceded by **lib** and has an extension of **.a**. Neither is included on the gcc command line. The **-l** options must always be placed after source code and object code filenames on the command line.

With multiple-file programs, you need to keep in mind the difference between the C compiler and the linker. The purpose of a C compiler is to generate object code, whereas the purpose of a linker is to build an executable file using object code files. The C compiler will individually compile each source code file, generating a separate object code file for each one. These object code files will have the extension **.o** instead of **.c**. You compile and link multiple file programs using the same **gcc** command. Simply list the source code filenames as arguments on the command line. In the next example, the user compiles the bookrecs program by invoking gcc with the source code files that make it up. The **-o** option specifies that the executable file will be called **bookrecs**.

```
$ gcc main.c io.c -o bookrecs
```

You can use the gcc utility to perform just a link operation by only listing object code files as its arguments. An object code file has a **.o** extension. In the next example, the user just performs a link operation. No compiling takes place. Of course, this operation assumes that the object code files have been previously generated.

```
$ gcc main.o io.o
```

As you develop and debug your program, you will be making changes to source code files and then recompiling your program to see how it runs. If you have a very large program made up of many source code files, it would be very inefficient to recompile all of them if you only made changes to just a few of them. Those to which you made no changes do not need to be recompiled, just linked. You can direct the gcc utility to do just that, by mixing source code files and object code files as arguments on the command line. Source code files have a **.c** extension, and object code files have a **.o** extension. gcc will compile the source code files you specified on the command line, but will only use the object code files with the linker. This has the advantage of being able to compile only those files where changes have been made. If changes were made in **main.c**, but not in **io.c**, **io.c** would not have to be compiled. You would then specify the source code file **main.c** and the object code file **io.o** on the command line. In the next example, **io.o** will not be compiled, whereas **main.c** will be compiled:

```
$ gcc main.c io.o -o bookrecs
```

ELF and a.out Binary Formats

There are two possible formats for binary files such as executable programs. The first is the **a.out** format that is the original format used on Unix systems as well as early Linux systems. The term "a.out" comes from the default name given to an executable file by the Unix C compiler. As shared libraries came into use, difficulties arose with the **a.out** format. Adapting an **a.out** format for use as a shared library is a very complex operation. For this reason, a new format was introduced for Unix System 5 release 4 and for Solaris. It is called the Executable and Linking Format (ELF). Its design allowed for the easy implementation of shared libraries.

The ELF format has been adopted as the standard format for Linux systems. All binaries generated by the gcc compiler are in ELF format (even though the default name for the executable file is still **a.out**). Older programs that may still be in the **a.out** format will still run on a system supporting ELF.

C++ and Objective C: g++

The gcc utility is also a C++ compiler. It can read and compile any C++ program. However, it will not automatically link with the C++ **Class** library. You would have to invoke it on the command line. Alternatively, you can use the command **g++**, which invokes the gcc compiler with the C++ **Class** library.

C++ source code files have a different extension than regular C files. Several different extensions are recognized for C++: C, cc, cxx, or cpp. Other than this difference, you compile C++ programs just as you would C programs. Instead of **gcc**, it is preferable to use the **g++** command. The following example compiles a C++ program, **myprog.cpp**:

```
$ g++ myprog.cpp -o myprog
```

The gcc compiler also supports Objective-C programs. Objective-C is an object-oriented version of C originally developed for NeXt systems. To compile a program in Objective-C, you use the **gcc** command with the **-lobjc** option, which links to the Objective-C library, **libobjc.so**.

Other Compilers: Pascal, Ada, Lisp, and Fortran

A great many programming languages are supported on your Linux system. Many are available on your OpenLinux CD-ROM. In addition to C and C++, you can compile Pascal, ADA, Lisp, Basic, and Fortran programs. In several cases, the compiling is handled by the gcc compiler, which is designed to recognize source code files for other programming languages. For example, G77 is the GNU Fortran compiler. This compiler is integrated with the gcc compiler. The command **g77** will compile a Fortran program by invoking the gcc compiler with options to recognize Fortran code, using the G77 features of gcc. The ADA 95 compiler is called gnat. The info file on ADA provides detailed information on gnat. You can compile an ADA program using the command **gnatmake** with the filename.

Creating and Using Libraries: Static, Shared, and Dynamic

There are usually functions used in a C program that rarely need to be compiled and are used repeatedly. There may also be functions that you may want to use in different programs. Often, such functions perform standardized tasks such as database input/output operations or screen manipulation. You can precompile such functions and place them together in a special type of object code file called a *library.* The functions in such a library file can be combined with a program by the linker. They save you the trouble of having to recompile these functions for each program you develop.

Different types of applications make use of specialized libraries that are placed in system directories and made available for use in developing programs. For example, there is a library, **libdbm**, that contains dbm functions for implementing database access to files. You can use these functions in your own programs by linking to that library. Mathematical applications would use the math library, **libm**, and X Window applications would use the Xlib library, **libX11**. All programs make use of the standard C library, **libc**, that contains functions to perform tasks such as memory management and I/O operations (a new version of the GNU **libc** library, 2.0, is now available). These libraries are placed within system directories such as **/usr/lib**, where they can be accessed by anyone on the system. You can also create your own library just for use with your own particular program, or make one that you would want to be accessed by others.

Libraries can be either static, shared, or dynamic. A *static* library is one whose code is incorporated into the program when it is compiled. A *shared* library, however, has its code loaded for access whenever the program is run. When compiled, such a program simply notes the libraries it needs. Then, when the program is run, that library is loaded and the program can access its functions. A *dynamic* library is a variation on a shared library. Like a shared library, it can be loaded when the program is run. However, it does not actually load until instructions in the program tell it to. It can also be unloaded as the program runs, and another could be loaded in its place. Shared and dynamic libraries make for much smaller code. Instead of a program including the library as part of its executable file, it only needs a reference to it.

Most libraries currently developed are shared libraries. Shared libraries were made feasible by the implementation of the ELF binary format, though there is an older **a.out** format for shared (tagged) libraries. ELF is currently the standard format used for all binary files in Linux.

The GNU libraries are made available under a Library GNU Public License (LGPL). The conditions of this license differ from the standard GNU license in that you are free to charge for programs developed using these libraries. However, you do have to make available the source code for those libraries you used.

Libraries made available on your system reside in the **/usr/lib** and **/lib** directories. The names of these libraries always begin with the prefix **lib** followed by the library name and a suffix. The suffix differs, depending on whether it is a static or shared library. A shared library has the suffix **.so** followed by major and minor version numbers. A static library simply has a **.a** extension. A further distinction is made for shared libraries in the old **a.out** format. These have the extension **.sa**.

```
libname.so.major.minor
libname.a
```

The *name* can be any string, and it uniquely identifies a library. It can be a word, a few characters, or even a single letter. The name of the shared math library is **libm.so.5**, where the math library is uniquely identified by the letter **m** and the major version is 5. **libm.a** is the static math library. The name of the X Windows library is **libX11.so.6**, where the X Windows library is uniquely identified with the characters X11 and its major version is 6.

You can link libraries to your programs using the gcc compiler. For example, the **libc.so.5** library contains the standard I/O functions. This library is automatically searched and linked whenever the linker generates an executable file. The standard I/O library contains numerous functions that include input/output operations such as **printf**. There are other system libraries that you can access, such as the math library. Though the **libc.so.5** library is automatically linked, most other system libraries need to be explicitly referenced on the command line.

Most shared libraries are found in the **/usr/lib** and **/lib** directories. These will always be searched first. Some shared libraries are located in special directories of their own. A listing of these is placed in the **/etc/ld.conf** configuration file. These directories will also be searched for a given library. By default, Linux will first look for shared libraries, then static ones. Whenever a shared library is updated or a new one installed, you need to run the **ldconfig** command to update its entries in the **/etc/ld.conf** file as well as links to it (if you install from an RPM package, this is usually done for you).

To reference a library file in one of these searchable directories when you invoke the gcc compiler, you use the **-l** option followed by the unique part of a system library's name: **-l***name*. To instruct the linker to use the standard math library, you enter **-lm** on the gcc command line. **-l** will look first for a **lib*name*.so** file, in this case, **libm.so**. This is a link to the actual library file. In the next example, the bookrecs program is created and linked to the math library. Notice the **-lm** option.

```
$ gcc main.c io.c -o bookrecs -lm
```

Many different libraries are currently available for your use. One of the more popular is the **libncurses.a** library, which contains simple cursor movement routines. You would reference the **libncurses.so** library on the command line with **-lncurses**. In the next example, the user invokes both the math and curses libraries:

```
$ gcc main.c io.c -o bookrecs -lm -lncurses
```

To reference a library in another directory, you have to specify that directory using the **-L***dir* option. This option adds the specified directory to the list of directories that will be searched with the **-l** option. In the following example, the user links to a library in the **mydir** directory called **myio.so**. For a shared library, you will first have to have the dl and ld link names set up, such as **libmyio.so** and **libmyio.so.1** for a **libmyio.so.1.0** file.

```
gcc main.c -o bookrecs -Lmydir -lmyio
```

The gdb Symbolic Debugger

The gdb program is the symbolic debugger available on your Linux system. If you run your program and it crashes for some reason, you can use a symbolic debugger to track down the error. A symbolic debugger allows you to step through your program line by line, displaying the source code for each line as you execute it. You can decide to stop in specific functions and display the contents of active variables. You can even check specific addresses and the contents of the stack.

Note Instead of using the command line interface shown here, you can use any of several available graphical interfaces for gdb. These include the GNU Data Display Debugger (DDD) available from Red Hat 7.2 on your CD-ROM and at **ftp.redhat.com**. Also available are the GNU Visual Debugger (GVD) from **www.gnome.org**, and the KDE Debugger, KDbg, from **apps.kde.com**.

To be able to use an executable file with a symbolic debugger, you need to compile and link your program using the **-g** option. In the next example, a program is compiled and prepared for the symbolic debuggers. Once you have a prepared executable file, you can then use it with the symbolic debugger.

```
$ gcc -g main.c io.c
```

You invoke the gdb debugger with the keyword **gdb** and the name of the executable file. In the next example, the name of the executable file is **a.out**:

```
$ gdb a.out
```

The **gdb** command will place you in the debugger, replacing the Linux prompt (**$**), with the gdb prompt (**gdb**). You run your program in the debugger by typing at the prompt the command **run**:

```
 (gdb) run
```

If your program has in it an **fopen** or **open** statement, it means it will be using a data file at some point in the program. If this is so, then gdb also needs to know the name of such a data file. When you type **run** in gdb to run your program, you must also supply the actual names of such data files:

```
 (gdb) run filename
```

When you are finished, leave the debugger with the quit command, **q** or **quit**:

```
 (gdb) quit
```

Most gdb commands have a single-letter equivalent consisting of the first letter of the command. Instead of entering the command **run**, you can enter just **r**. For **quit**, you can enter the letter **q**, for **print** just **p**, and for **next** the letter **n**. The gdb commands are listed in Table 3.

Table 3: The: gdb Symbolic Debugger

Running Programs in gdb	**Command**	**Description**
r	**run**	Run the program.
q	**quit**	Quit gdb.

Table 3: The: gdb Symbolic Debugger

Running Programs in gdb	**Command**	**Description**
Displaying Variables and Arguments		
p *var*	**print** *var*	Display the contents of a variable.
p &*var*	**print &***var*	Display the address of a variable.
set *var* = *value*		Assign a value to a variable during the gdb session.
where		Display a stack trace showing a sequence of function calls with function names and their arguments.
	info locals	Display defined variables and arguments.
Displaying Lines		
l *linenum*	**list** *linenum*	Display lines beginning with the specified line number.
l *func*	**list** *func*	Display lines in a function.
l *num***,**num	**list** *num***,**num	Display a range of lines.
Stepping and Continuing Execution		
n	**next**	Single-step execution line by line, executing the current line and displaying the next line to be executed.
s	**step**	Single-step execution line by line, executing the current line and displaying the next line to be executed.
c	**cont**	Continue execution of the program.
Setting and Deleting Breakpoints		
b	**break**	Set breakpoint at current line.
	break *line*	Set breakpoint at specified line.
	break *func*	Set breakpoint at first line in the specified function.
	info break	List all breakpoints.
d *num*	**delete** *num*	Delete breakpoints. You need to specify the number of the breakpoint.
	delete	Delete all breakpoints.

You display the contents of a variable using the **print** command. Enter the word **print** followed by the variable name. In the next example, the user displays the contents of the **count** variable:

```
(gdb) print count
```

With the **where** command, you can display the function names and arguments of the functions that have been called at any point in your program. In the next example, the user is currently in the **calc** function. Entering the **where** command displays the functions **main**, as well as **calc** and its arguments.

```
 (gdb) where
#3 calc(newcost = 2.0) at calc.c:25
#1 main () at main.c:19
#2 0x8000455 in ___crt_dummy__ ()
```

You can obtain a listing of all the variables and arguments defined in a function. The **info locals** command will display variable and argument values currently defined. In the next example, the user displays the defined variables:

```
 (gdb) info locals
cost = 2
name = "Richard\000\000"
count = 10
count2 = 10
nameptr = 0x8000570 "petersen"
countptr = (int *) 0xbffffde8
```

You can set breakpoints in your program using the **break** command. When you reach a breakpoint, your program will stop. You can then step through your program line by line using the **next** or **step** command. When you wish, you can advance to the next breakpoint by using the **cont** command.

Programming Tools

Many tools are available to help you prepare and organize your source code. The indent utility will indent the braces used for blocks in a consistent format, making the code easier to read. cproto generates a list of function declarations for all your defined functions for use in header files. f2c and p2c can translate Fortran and Pascal programs into C programs. xwpe is an X Windows programming environment similar to Turbo C. Many more tools are also available.

Once you have finished developing your software, you may then want to distribute it to others. Ordinarily, you would pack your program into a tar archive file. People would then download the file and unpack it. You would have to have included detailed instructions on how to install it and where to place any supporting documentation and libraries. If you were distributing the source code, users would have to figure out how to adapt the source code to their systems. There are any number of variations that may stop compilation of a program.

The Red Hat Package Manager (RPM) and Autoconf are designed to automate these tasks. Autoconf is used to automatically configure source code to a given system. The Red Hat Package Manager will automatically install software on a system in the designated directories, along with any documentation, libraries, or support programs. Both have very complex and powerful capabilities, making them able to handle the most complex programs. Several Linux distributions like Red Hat and Caldera support RPM packages.

Development Tools

An application is an executable program created by a programmer using one of several programming languages. Linux provides several utilities with which a programmer can control development of an application. Foremost among these is the make utility. The make utility interfaces with the Linux operating system to provide an easy way to maintain and compile programs. The RCS utility allows you to better control program changes. It organizes changes into different versions that can be stored and later accessed. You can even use the man utility to create your own online documentation for your applications. All of these utilities are complex and powerful tools.

The make Utility

You will often be working with a program that has many source code files. As you develop the program, making modifications, you will need to compile the program over and over again. However, you need only compile those source code files in which you made changes. The linker then links the newly generated object code files with previously compiled object code files, creating a new executable file. The fact that only a few of your source files are actually compiled drastically cuts down on the work of the compiler. Each time you need a new executable program, you do not need to recompile each source code file.

It can be very difficult in large programs with many source code files to keep track of which files have been changed and need to be compiled and which files need only to be linked. The make utility will do this for you. make was designed for a development environment in which different source code files in a program are constantly being modified. make keeps track of which source files have been changed and which have not. It then recompiles only those that have been changed, linking them with the rest of the object code files to create a new executable file. In the next example, the user enters the command **make** on the command line to invoke the make utility. make then compiles those files that have recently been modified and creates a new executable file. make displays each Linux command it executes.

```
$ make
gcc -c main.c
gcc -c io.c
gcc main.o io.o
```

To understand how the make utility works, you need to realize that it uses a source code file's time stamp to determine whether or not it should be compiled. Whenever a file is created, re-created, or modified in any way, a new time stamp is placed on it by the Linux operating system. If you create a file at 1:00, that file is stamped with the time 1:00. If you then change the file at 6:00, the file is restamped with the time 6:00. When you are compiling a program, only those source code files that have been changed need to be recompiled. Since a change in any file changes the time stamp, the time stamp can be used to determine which files need to be compiled. In this way, make knows which files need to be compiled and actually selects the files to be compiled for the programmer.

A dependency line specifies a dependency relationship between files. make operates in terms of dependencies. A source code file is used to create an object code file, which in turn is used to create a runnable program. The program can be said to be dependent on the object code file, which in turn is dependent on the source code file. You need to specify the dependency relationship between a source code file and an object code file in a dependency line. In another dependency line, you need to specify the dependency relationship between an executable file and all its object code files.

A dependency line can be thought of as a kind of conditional statement. The dependency relationship is its test condition. If an object code file depends on a source code file and the source code file has been recently modified, then the test condition is true and the file is then recompiled. However, the syntax for a dependency line is a bit more complex than a standard conditional statement. A dependency line consists of three components: a target file, a list of dependency files, and a Linux command. If any of the dependency files have been modified more recently than the target file, the Linux command is executed. The target file and the dependent files are written on the same line, separated by a colon. You can either place the Linux command on the same line, separated from the dependent files by a semicolon, or you can place the Linux command on the next line preceded by a tab. You can list more than one Linux command if you wish. When entering Linux commands on the same line, you separate them with semicolons. Entered on a separate line, each Linux command has to be preceded by a tab. The dependency line ends with a following empty line. In these examples, the Linux command is an invocation of the gcc compiler, compiling a source code file or linking object code files. The syntax for a dependency line is as follows:

```
target file : dependent files ; Linux command
empty line

target file : dependent files
tab   Linux command
empty line
```

In the following **Makefile**, we construct the dependency lines for a C program consisting of two source code files: **main.c** and **io.c**. In such a two-file program, there are really five files to manage. For each **.c** file there is a corresponding **.o** file. There is the executable file, **a.out**. You need to set up your **Makefile** with dependency lines to manage all of these files, specifying dependencies for each. An object code file (**.o**) is dependent on a source code (**.c**) file. An executable file, **a.out**, is dependent on several object code files (**.o**). In the example, **a.out** is dependent on (made up of) the two object code files **main.o** and **io.o**. Each object code file is, in turn, dependent on its respective source code files: **main.o** on **main.c**, and **io.o** on **io.c**.

In the **Makefile**, three dependency lines are needed for the **a.out**, **main.o**, and **io.o** files, respectively. Notice that the linking and compilation of the program are split up among the different dependency lines. The Linux command for the **a.out** target only links the two object code files, creating a new executable file. It invokes gcc with only object code files (**.o**), causing only the linker to be invoked. The Linux commands for the **main.o** and **io.o** targets only compile, creating **.o** object files. The **-c** option used with **gcc** means that no linking is done, only compilation, generating the object code file for this source code file.

makefile

```
a.out : main.o io.o
gcc main.o io.o

main.o : main.c
gcc -c main.c
```

```
io.o : io.c
gcc -c io.c
```

You can also create dependency lines for special operations such as printing files or installing an application. Many software packages that you create with make will also have an install target to install the application program on your system.

```
$ make install
```

The Revision Control System: RCS

When you work on a major project, you are continually changing source code. You may detect bugs, or you may add other features. Sometimes changes may unintentionally result in new bugs. A record of all the changes to your program may help you track down bugs and any possible design errors. The Revision Control System (RCS) is a Linux utility that keeps track of all changes that you have made to a program. In effect, RCS provides you with a set of prior versions for your program. You can view each version and examine the changes made for each.

RCS is very helpful in a situation in which a team of programmers is working on the same program. Each programmer may make changes to the program at different times. RCS can record each change a programmer makes and when it was made. It can even include notes about a change.

RCS stores an original version of a file and then records all changes to the file. Using this information, it can generate any one of several possible versions of a file. RCS does not actually store separate full versions of a file. Instead, it uses the change information and the original file to create a full version of the file. The commands that manage RCS files are listed in Table 4.

Table 4: The: RCS Utility

Commands		**Description**
RCS		The Revision Control System (RCS) allows you to control the development of a program. With RCS, you can establish different versions of your program as you make changes. You can later retrieve different versions or obtain a record of how your program was developed. An RCS file is created with the **ci** command and managed with the **rcs** command. Versions are retrieved with the **co** command, and versions are erased with the **rcs -o** command.
ci		The **ci** command updates an RCS file, creating new versions. If the RCS file does not already exist, **ci** will create it using the extension **,v**. You usually use **ci** to save an edited copy of a file that you previously retrieved using **co** with the **-l** option. Saving this edited copy will create a new version for the file within the RCS file.

Table 4: The: RCS Utility

Commands		**Description**
		$ ci main.c
	-r*version*	This option allows you to specify the release and version number you want to begin with when creating a new version. **$ ci -r5.2 main.c**
co		The **co** command retrieves a version of an RCS file. With no option, it retrieves a read-only version. If no version number is specified, the most recent version is retrieved.
	-l	This option retrieves an editable version of an RCS file, locking the file to prevent others from changing it. The file remains locked until you use **ci** to check it back in. **$ co -e main.c**
rcs		The **rcs** command manages RCS files and can be used to control access to an RCS file by other users.
	-a*user-name*	This option will add *user-name* to the list of users that can access a specified RCS file. **$ rcs -arobert main.c**
	-e*user-name*	This option will remove *user-name* from the list of users that can access a specified RCS file. **$ rcs -erobert main.c**
	-l*release*	This option will lock a specific release for everyone but the creator of the file. **$ rcs -l2 main.c**
	-u*release*	This option will unlock a specific release. **$ rcs -u2 main.c**
	-L*release*	This option will lock a specific release for everyone, including the creator of the file. **$ rcs -L2 main.c**
	-U*release*	This option will unlock a specific release for everyone, including the creator of the file. **$ rcs -U2 main.c**
	-o*release*	This option will delete a version from an RCS file. **$ rcs -o2.3 main.c**
rlog		The **rlog** command outputs information about the different releases and versions in a RCS file. Without an argument, it outputs summary information for each version.
	-r*version*	This option will output information about a specific version. **$ rlog -r2 main.c**
	-d*date*	This option will output information about versions created on a specified date. The format for the date is year, month, day separated by slashes, and hour, minute, second

Table 4: The: RCS Utility

Commands		Description
		separated by colons. All except the year are optional. $ **rlog -d93/04/12 main.c**
	-d<	Followed by a less-than sign, this option will output information that is earlier than a specified date. **$ rlog -d93/04/12 main.c**
	-d<	Preceded by a greater-than sign, this option will output information that is later than a specified date. **$ rlog -d93/04/12 main.c**

A set of recorded changes to a file is called a *version.* Each version is assigned a version number that has several components, the first two of which are the release and level numbers. By default, the first version is assigned a release number of 1 and a level number of 1. A version is often referred to by its release and level numbers. The first version is called version 1.1 or delta 1.1. Subsequent versions will have the same release number with an incremented level number. The next version will be 1.2, then 1.3, etc. You can also change the release number manually.

To create an RCS file, you first create an RCS directory. Within this directory are placed the RCS files for your programs. You can then create an RCS file with the **ci** command. The **ci** command (which stands for "check in") takes one argument, the name of the original file. The RCS file will be created in the RCS directory with the extension **,v**. A **main.c** file will have an RCS file called **main.c,v** in the RCS directory. If your program is initially made up of several source code files, you need to create an RCS file for each one, including its source code suffix. In the next example, the user creates an RCS file for the **main.c** program:

```
$ ci main.c
RCS/main.c,v main.c
enter description, terminated with single '.' or end of file:
NOTE: This is NOT the log message!
>> Bookrecs main program
>> .
initial revision: 1.1
done
```

To edit your source code file using RCS, you must first have RCS generate a copy of the source code file. This copy will have the same name as the RCS file, but without the **,v** suffix. For the **main.c,v** file, RCS will generate a file called **main.c**. To save the copy once you have made changes, you simply register any changes you make to the RCS file.

The RCS **co** command (which stands for "check out") generates a copy of the source code file. The **co** command has several options. The **co** command with no options simply generates a read-only copy of the source code file. The **-l** option generates a copy of the source code file that you can edit; **-l** stands for lock, and when you use this option, the **main.c** program in the RCS **main.c,v** file is locked. The locking mechanism used by RCS permits only one programmer at a time to change a given file. When you have finished your modifications, you "check in" the program code, registering your changes and unlocking it for use by others. In the next example, the **co** command generates an editable copy of the source code file **main.c,v**:

```
$ co -l main.c
RCS/main.c,v --> main.c
revision 1.1 (locked)
done
```

Once you have finished editing your source code file, you then register your changes in the RCS file with the **ci** command. You enter the keyword **ci** followed by the name of the RCS file. You are then prompted to enter comments. In effect, editing a copy of the file generated with **co** creates a new version of the source code file, a new set of changes constituting a new version. The new version number (1.2) is displayed. In the next example, the user saves the changes to **main.c** by generating a new version, 1.2:

```
$ ci main.c
RCS/main.c,v main.c
new revision: 1.2; previous revision: 1.1
enter log message, terminated with single '.' or end of file:
>> Added prompts
>> .
done
```

The Concurrent Versions System: CVS

The Concurrent Versions System (CVS) has become a popular alternative to RCS, primarily because it is easy to use for program development over the Internet. CVS allows several developers to work on a file at the same time. This means that CVS supports parallel development, where programmers around the world can work on the same task at the same time. Developed originally as a front end to RCS, CVS shares many of the same commands (see Table 5). You can find out more about CVS from **www.cvshome.org**.

Table 5: CVS: Commands and Options

Commands		**Description**
commit,ci		The **commit** command updates a CVS file, creating new versions.
	r*version*	This option allows you to specify the release and version number you want to begin with when creating a new version.
checkout,co		The **checkout** command retrieves a version of a CVS file.
	-D *date*	Check out revisions for a specific date.
	-d *dir*	Check out to a specified directory.
	-r *version*	Check out a specified version.
import		Import files to CVS repository.
login		Log into a remote CVS repository.
logout		Log out from a remote CVS repository.
rdiff		Show the differences between releases.
remove		Remove an entry from a repository.
update, up		Update from a repository.

Table 5: CVS: Commands and Options

Commands		Description
init		Create a CVS repository.
admin		Administer files in the repository.
	-l*release*	Lock a revision.
	-u*release*	Unlock a revision.
	-o*release*	Delete revision from the repository.

Note You can use CVS GUI clients on Gnome and KDE to manage your CVS repositories or access those on the Internet. For Gnome you can use Pharmacy, and for KDE you can use Cervisia or LinCVS.

CVS is structured along the same lines as RCS, using check-in and check-out procedures for projects. To use CVS, you first create a directory to hold your project files and then use the import option to install your project files there. In the following example, the **/home/projects** directory is designated as the CVS repository directory. You use the command **cvs** with the **-d** option and the name of the directory along with the **init** option.

```
$ cvs -d /home/projects init
```

You can set the CVSROOT shell variable to the path of the repository directory and then export the variable. The **cvs** command, whenever executed, will automatically check this variable for the location of the repository.

```
CVSROOT=/home/projects
export CVSROOT
```

To install files for an already existing project in the repository directory, you use the **import** option. You can install them to a subdirectory in the repository. A CVS repository can support multiple projects, each in its own subdirectory. You will also need to provide arguments specifying the supplier and the release. First you change to the directory that holds your project files, then you issue the **cvs** command with the **import** option. The following example will import all the files in the current working directory—in this case, **myproject-so-far**—to the **myproject** directory in the CVS repository (**/home/projects/myproject**). The supplier is **richp**, and the release is the first release, as indicated by the **start** argument. If the current directory holds the file **main.c**, the **main.c** file will be installed in the **/home/projects/myproject** directory along with any other files in the current directory.

```
$ cd myproject-so-far
$ cvs import myproject richp start
```

Whenever you install projects, or add, change, or remove files in a CVS repository, you need to supply a log message describing the action. CVS will automatically open your standard editor to let you enter the description. Alternatively, you can use the **-m** option to include the log message in the command. Be sure to quote the message string. You would use the editor for multiline detailed messages, and the **-m** option for short one-line messages. The following example includes a log message with the **import** command:

```
$ cvs import –m "Install myproject into CVS" myproject richp start
```

To work on a file, you must first retrieve it from the repository using the check-out option, **co**. You then work on it in a project directory that will be created as a subdirectory in your local directory. When you are ready to check the file back into the repository, you use the **commit** option. The following example will extract the **main.c** file from the CVS repository for **myproject**:

```
$ cvs co myproject/main.c
$ cd myproject
```

You then change to the subdirectory called **myproject** that has been created in the current working directory. There you will find the extracted version of **main.c**. You can then edit and change **main.c**. To check it back into the CVS repository, you would use the **commit** option.

```
$ cvs commit -m "Modified main.c" main.c
```

To see the changes, use the **diff** option. The **rdiff** option lets you see any changes to the entire release.

```
$ cvs diff
```

To check out the entire project, you use the project name. All the files for the project will be extracted to a subdirectory with that project name.

```
$ cvs co myproject
$ cd myproject
```

You can then work on any of the files and then check in the entire project when you are finished.

```
$ cvs commit -m "Major changes to myproject" myproject
```

As you work on a project with numerous files, you can use the update option, **up**, to check out just the ones you need.

```
$ cvs up myproject
```

To add a new file, you use the **add** option, and to delete a file you use the **remove** option. You effect the changes with the **commit** option.

```
$ cvs add mynewfile
$ cvs commit -m "Added mynewfile" myproject
```

To create and extract new releases, you use the **–r** options and specify the release number, such as 1.2 or 3.5. You create a new version with the **commit** command. The following example creates a new version, 1.2:

```
$ cvs commit -m "Created release 1.2" –r 1.2 myproject
```

You can then check out that release with the **co** option:

```
$ cvs co –r 1.2 myproject
```

To access a repository on the Internet, you simply specify as the repository root the Internet site and the remote directory for that repository. You can do this by assigning the repository to the CVSROOT shell variable. The format for specifying the remote repository is as follows:

```
:method:user@hostname:/path/to/repository
```

The following example assigns the repository for KDE to the CVSROOT variable:

```
export CVSROOT=:pserver:anonymous@anoncvs.kde.org:/home/kde
```

To access the remote repository, you first log in with the **login** option:

```
$ cvs login
```

Then you use the standard **co** (check-out) and **commit** options to check out and check in projects. When doing so, you may want to use the compression options such as **-z4** to speed transmission time.

Note If you want to set up a repository on your own system that is accessible over the Internet, you need to install and configure the CVS server.

Autoconf

If you were distributing the source code, users would have to determine how to adapt the source code to their systems. Any Unix or Linux system can compile any software written in the C programming language. Different Unix and Linux systems have different configurations, however, some using different compilers or placing programs and libraries in different system directories. Different types of support libraries may be present. In the past, to compile software on different systems, the software had to be configured manually for each system. For example, if your system has the gcc compiler instead of the cc compiler, you would have to set that feature in the software's **Makefile**.

The Autoconf program is designed to automate the configuration process. It automatically detects the configuration of the current Unix system and generates an appropriate **Makefile** that can then be used to compile that software on this particular system. Much of the current software on the Internet in source form uses Autoconf. A detailed manual on Autoconf can be found in the **/usr/info** directory and is called **autoconf.info.** You can use the **info** command to view it. (You can also view the text with any text editor.) The general operations are described here.

Software that uses Autoconf performs the configuration without any need of the actual Autoconf software. Special shell scripts included with the software detect the different system features the software needs. The **./configure** command usually automatically configures the software for your system. As the configuration is performed, it checks for different features one by one, displaying the result of each check. The operation is entirely automatic and doesn't even require the identity of the system on which it is working.

To create a configuration script for your own software, you use special Autoconf commands. The Autoconf applications package is available on your Red Hat CD-ROM. Generating the configurations involves several stages, using several intermediate configuration files.

Autoconf has many options designed to handle the requirements of a complex program. For a simple program, you may only need to follow the basic steps.

The goal is to create a **configure** script. Two phases are in this process, using the **autoscan** and **autoconf** commands. The first phase creates a **configure.scan** file, using the **autoscan** command. The **autoscan** command is applied directly to your source code files. Next, check the **configure.scan** file for any errors with an editor, make any changes or additions you want, and then rename it as the **configure.in** file. This file is used as input for the **autoconf** command, which then generates the **configure** script. The **autoscan** step is an aid in the creation of the **configure.in** file, but **autoscan** and the **configure.scan** file it generates are optional. You can create your own **configure.in** file, entering various Autoconf macros. These are described in detail in the Autoconf info file.

In addition, you need to create a version of the **Makefile** for your program, named **makefile.in**. This is essentially your original **Makefile**, with reference to special Autoconf variables. When the software is compiled on another system, the **configure.in** file detects the system's features, and then uses this information with the **makefile.in** file to generate a **Makefile** for that particular system. This new **Makefile** is then used to compile the program.

Autoconf is designed to create values for different features, which can then be used with **makefile.in** to create a new **Makefile** containing those features. The feature values are placed in special shell variables called *output* variables. You should place references to these shell variables in the **makefile.in** file wherever you want to use these values. For example, the CC variable holds the name of the C compiler on your system (cc or gcc). The **AC_PROG_CC** macro in the **configure** script detects the C compiler in use and places its name in the CC variable. A reference to this variable should be placed in the **makefile.in** file wherever you invoke the C compiler. The variable name is bounded by two @ symbols. For example, **@CC@** in **makefile.in** references the CC variable and its value is substituted in that place.

Once you have the **configure** file, you no longer need the **configure.in** file. You only need **configure**, **makefile.in**, and the source code files along with any header files. The **configure** file is a shell script designed to execute on its own.

Once another user has received the software package and unpacked all the source code files, you only need to take three steps: configuration, compilation, and installation. The **./configure** command generates a customized **Makefile** for the user's system, the **make** command compiles the program using that **Makefile**, and the **make install** command installs the program on the user's system.

```
./configure
make
make install
```

Online Manuals for Applications: man

As you develop a program for use on Linux, you may need to document it. If it is a large application worked on by several programmers at once, documentation may become essential. Documentation often takes the form of a manual describing different commands and features of a program. Many times an application is broken down into separate programs. It is very helpful to both users and program developers to be able to retrieve and display appropriate sections of an application's manual. Such an online manual provides instant access to documentation by all users and developers. The **man** command provides access to an online manual for Linux commands. You can also use the **man** command to manage your own online manual for your own applications. You can create Man documents using special Man text processing macros. You can then instruct the **man** command to read these documents.

One of the more common uses today of gnroff is the creation of online manual documents. The online manual that you find on your Linux system is actually a gnroff document. You use a set of macros called the Man macros to create your own online manual entries. If you create a new command and want to document it, you can create a manual document for it and read this document with the **man** command.

When you call the **man** command with the name of a document, the **man** command uses gnroff to format the document and then display it. The actual document that you create is an unformatted text file with the appropriate Man macros. The Man macros, which are very similar to the ms macros, are described in the next section. You can actually format and display a manual document directly by using the **gnroff** command with the **-man** option. In the next example, both commands display the manual document for the **ls** command.

```
$ man ls
$ gnroff -man /usr/man/man1/ls | more
```

Man Document Files

You can create a Man document file using any standard text editor. The name that you give to the text file is the name of the command or topic it is about with a section number as an extension. The name of a document about the **who** command would be called **who.1**. In the example described here, the document is about the **bookrec** command and has a section number **1**. Its name is **bookrec.1**.

A Man document is organized into sections with a running title at the top. The sections may be named anything you wish. By convention, a Man document for Linux commands is organized into several predetermined sections such as NAME, SYNOPSIS, and DESCRIPTION. You are, however, free to have as many sections as you want and to give them names of your own choosing. The actual document that you create is an unformatted text file with the appropriate Man macros. A manual document requires at least two different Man macros: **TH** and **SH**. **TH** provides a running title that will be displayed at the top of each page displayed for the document as well as the document's section number. You use the **SH** macro for each section heading. You can add other macros as you need them, such as a **PP** macro for paragraph formatting or an IP macro for indented paragraphs. The Man macros are listed in Table 6.

Table 6: Man: Command and Macros for Creating Online Documents

Commands and Options	**Description**
man	The manual command can search and display online manual documents. You can create your own online manual documents and instruct man to search for them. The **man** command searches for documents in section directories specified by the word "man" followed by a section number, **man1**. When you create your own manual directories, be sure to include section directories.

Table 6: Man: Command and Macros for Creating Online Documents

Commands and Options	**Description**
man *command-name*	Search for and display online manual documents. **man who**
MANPATH	Shell special variable that holds the directory pathname the **man** command automatically searches. You can add new pathnames with an assignment operation. **$ MANPATH=$MANPATH:/$HOME/man**
Options	
num	Search only specified section for a manual document. **$ man 3 bookrec**
-M *directory-name*	Search only specified directory for manual documents. **$ man –M $HOME/man bookrec**
The Man Macros	
.TH *title sec-num*	Used to enter running title of the online document.
.SH *section-name*	Used to enter section headings.
.B *word*	Used to boldface words such as command names.
.I *word*	Used to underline words.
.IP *option*	Format indented paragraph. Used to enter in options in the OPTIONS section.
.PP	Formats paragraph. Starts a new paragraph.

You enter a macro in your document at the beginning of a line and preceded by a period. Any text that you enter in the lines after the macro will be formatted by it. A macro stays in effect until another macro is reached. Some macros, like **SH** and **TH**, take arguments. You enter in arguments after the macro on the same line. The **SH** macro takes as its argument the name of a section. You enter the section name on the same line as the **SH** macro. The body of the section then follows. The body of the section text is entered in as a series of short lines. These lines will later be formatted by Man into a justified paragraph. In the next example, the user enters in the **SH** macro and follows it with the section name DESCRIPTION. In the following lines, the user enters in the text of the section:

```
.SH DESCRIPTION
.I bookrec
 allows the user to input a title and price
for a book. Then both elements
of the record are displayed
```

Your Man document is organized into a series of section heading macros with their names and text. The template here gives you an idea of how to organize your Man document. It is the organization usually followed by the online manual for Linux commands.

```
.TH COMMAND Section -number

.SH NAME
command and brief description of function
.SH SYNOPSIS
```

```
command and options. Each option is encased in brackets. This section
is sometimes called the SYNTAX.
.SH DESCRIPTION
 Detailed description of command and options. Use paragraph macros for
new paragraphs: PP, LP, and IP
.SH OPTIONS
 .Options used for the command
.SH EXAMPLES
 .Examples of how a command is used
.SH FILE
 .File uses by the command
.SH "SEE ALSO"
 References to other documentation or manual documents
.SH DIAGNOSTICS
 Description of any unusual output.
.SH WARNINGS
 Warning about possible dangerous uses of the command
.SH BUGS
 Surprising features and bugs.
```

Within the text of a section, you can add other macros to perform specific text processing operations. You enter in these macros on a line by themselves. Some will also take arguments. The **.PP** macro starts a new paragraph. The **.IP** macro starts an indented paragraph and is usually used to display options. You enter in the option as an argument to the **.IP** macro and then the following text is indented from it. The **.I** macro underlines text. The **.B** macro will boldface text. Both the **.B** macro and **.I** macro take as their argument the word you want to boldface or underline. By convention, the command name and options in the NAME section are set in boldface with the **.B** macro. Any other use of command names is usually underlined with the **.I** macro. If you use any hyphens, you need to quote them with a backslash.

In the next example, the user creates an online document for the bookrec program described earlier. Notice how the text is formatted into justified paragraphs. The options are displayed using indented paragraphs specified by the **.IP** macro.

bookrec.1 - man document file man output

```
.TH BOOKREC 1
.SH NAME
bookrec \-Input and display a book record
.SH SYNOPSIS
.B bookrec
[ \-t ] [ \-p] [ \-f]
.SH DESCRIPTION
.I bookrec
 allows the user to input a title and price
for a book. Then both elements
of the record are displayed
.SH OPTIONS
.IP t
Display only the title
.IP p
Display only the price.
.IP f
Save the record to a file
.SH FILES
 The command uses no special files.
.SH "SEE ALSO"
 printbook (1)
```

```
.SH DIAGNOSTICS
 Date output has the form of m/d/y.
.SH BUGS
 The program can only read and display one record.
.br
 It does not as yet allow you to read records from a file.
```

The man utility looks for a particular manual document in a system directory set aside for manual documents such as **/usr/man**. The **/usr/man** directory itself does not contain manual documents. Rather, it contains subdirectories called section directories, which in turn contain the documents. The name of a section directory begins with the word "man" and ends with the section number. **man1** is the name of the first section directory in a manual. There are usually about seven section directories, beginning with **man1** and ending with **man7**. In your own manual directory, you can have as many or as few section directories as you want, though you always have to have a **man1** directory.

Section directories allow you to create several documents of varying complexity and subject matter for the same command or topic. For example, the document in section 1 for the **man** command gives only a general description, whereas the document in section 7 for the **man** command lists all the Man macros. These documents are identified by section numbers and reside in the appropriate section directory. The section number of a document, as noted previously, is entered in as an argument to the **.TH** macro. To retrieve an online document from a particular section, enter in the section number before the command name. The next example displays the document on the **bookrec** command that is in the third section directory. If you enter no section number, the first section is assumed.

```
$ man 3 bookrec
```

Web Chapter 43: Gnome and KDE Programming

Overview

Development support for application interfaces was an integral part of both the Gnome (GNU Network Object Model Environment) and KDE (K Desktop Environment) projects from their inception. Any application can be designed easily to use either Gnome or KDE graphical components such as windows, toolbars, and buttons. Their aim is to provide not just a consistent interface, but also a flexible platform for the development of powerful applications. Both are completely free under the GNU Public License, with no restrictions.

Both Gnome and KDE provide extensive development libraries that contain functions and definitions you can use in C and C++ programs to create Gnome or KDE interfaces for your applications. You can code these directly into your program source code, or use a GUI development tool like KDevelop or Glade to construct your interfaces by simply selecting and combining different components. This chapter will briefly describe essential aspects of both Gnome and KDE development. For more information and detailed documentation, check the Gnome development site at **developer.gnome.org**, and the KDE development site at

developer.kde.org. In addition, numerous open source projects for both Gnome and KDE are under development at **sourceforge.net**. You can check here to download and examine source code for the kind of projects you may be interested in. You can even start your own project there. Table 1 lists several Gnome and KDE development sites resources.

Table 1: Gnome: and KDE Development Resources

Site	Description
developer.gnome.org	Gnome development site, including detailed documentation
glade.gnome.org	Glade, the GTK+ User Interface Builder
developer.kde.org	KDE development site, including detailed documentation
www.kdevelop.org	KDevelop Integrated Development Environment (IDE) for KDE applications
www.trolltech.com	QT site with documentation for QT libraries used in KDE, as well as for QT Designer
www.sourceforge.net	Open source development site for many Gnome and KDE projects

Gnome Programming: Glade

Gnome provides libraries of Gnome GUI tools that developers can use to create Gnome applications, and programs can be said to be Gnome compliant if they use buttons, menus, and windows that adhere to a Gnome standard. Gnome applications make use of Gnome, GTK Toolkit, GGTK Drawing Kit, and GNU libraries. For detailed descriptions of the functions, definitions, and structures contained in these libraries, it is strongly recommended that you use the extensive documentation available on the Gnome developer's Web site at **developer.gnome.org**. The Documentation section includes detailed tutorials, manuals, and reference works, including the complete reference for the Gnome, GTK, and GDK APIs.

GTK+ is the widget set used for Gnome applications, and its look and feel was originally derived from Motif. A widget set is the set of GUI objects that are available for use in a desktop. Buttons, windows, and toolbars are all examples of widgets. The widget set is designed from the ground up for power and flexibility. For example, buttons can have labels, images, or any combination thereof. Objects can be dynamically queried and modified at run time. GTK+ also includes a theme engine that lets users change the look and feel of applications using these widgets. At the same time, the GTK+ widget set remains small and efficient.

The GTK+ widget set is entirely free under the Library General Public License (LGPL). The LGPL allows developers to use the widget set with proprietary as well as free software. The widget set also features an extensive set of programming language bindings including C++, Perl, Python, Pascal, Objective-C, Guile, and Ada. Internalization is fully supported, permitting applications based on GTK+ to be used with other character sets, such as those of Asian languages. The drag-and-drop functionality supports both XDND and Motif protocols, allowing drag-and-drop operations with other widget sets that support these protocols, such as Qt and Motif.

Programs written to work on Gnome are essentially C programs that contain Gnome and GTK+ functions. The Gnome and GTK+ functions handle the Gnome desktop operations for a program. When programming for Gnome, you will make use of a very extensive set of functions and structures contained in many libraries—these functions and structures make up the different components that go into a Gnome application.

This chapter can only provide a general overview of these libraries and of how you use them to create Gnome programs. Though Gnome is not as easy to use as Tk, programming in Gnome requires the use of only a few basic functions to create simple user interfaces. You can think of GTK+ functions as lower-level operations and Gnome functions as easy-to-use higher-level operations. The Gnome functions usually incorporate several GTK+ functions, making GUI tasks relatively easy to program. A Gnome program is essentially a C program with GTK+ functions as well as Gnome functions. Because several basic Gnome operations are handled by GTK+ functions, this chapter begins by discussing basic GTK+ programming, and then discusses Gnome programs.

The Gnome libraries provide the highest-level functions used in Gnome applications. Below them are the GTK+ libraries. GTK+ is the toolkit developed for the GNU Image Manipulation Program (GIMP). GTK+ is made up of the GIMP Toolkit (GTK) and GIMP Drawing Kit (GDK) libraries.

GTK contains the functions and structures for managing widgets and user interface tasks. These functions and structures can be accessed directly in any Gnome program. In fact, a Gnome application is a GTK program with Gnome library functions. GTK functions and structures are C++ program objects, both designed to be used in a C++ style program.

GDK contains lower-level functions that are used to connect GTK to the **Xlib** libraries. The **Xlib** libraries hold functions that perform the actual X Window System operations. Both GTK and Gnome also make use of the standard C functions provided by the **Glib** library. Table 2 lists the different Gnome components.

Table 2: Gnome: Components

Gnome Component	**Description**
Gnome libraries	Contains high-level Gnome functions
GTK (GIMP Tool Kit)	Contains widgets and GUI functions
GDK (GIMP Drawing Kit)	Provides a low-level wrapper for **Xlib**
Xlib	Provides X Window operations
Glib	Contains the GNU C library of standard functions

Gnome applications also make use of ORBit and **Imlib** (Image Library). Gnome uses the Common Object Request Broker Architecture (CORBA), which allows software components to interconnect, regardless of the computer language in which they are implemented or the kind of machine they are running on. The Gnome implementation of CORBA is called ORBit. With ORBit, programs can locate and request services from an object, even one located across a network. For example, an editor could request the use of a spreadsheet. **Imlib** contains functions for managing images in various formats, letting you display, scale, save, and load images into your program.

Gnome Libraries

The Gnome libraries make it possible for Gnome applications to have the same kind of GUI interface with the same look and feel. Though a Gnome application is a GTK program with Gnome library functions, the Gnome library provides several complex higher-level widgets, as well as many simple operations not included in the GTK+ libraries. Table 3 lists the Gnome libraries.

Table 3: Gnome: Libraries

Library	**Description**
libaudiofile	Reads a wide variety of audio file formats (AIFF, AIFC, WAV, and NeXT/Sun au).
libgdk_imlib	Includes functions to load multiple file formats (JPEG, GIF, TIFF, PNG, XPM, PPM, PGM, PBM, and BMP).
libgtk	This is the GTL Toolkit library. Gnome applications are written entirely using **libgtk** for all GUI elements (buttons, menus, scroll bars, and so on).
libgnome	Includes utility routines for the Gnome desktop environment, such as routines for configuration, help, managing mime types, and managing sessions. This library is independent of any GUI toolkit.
libgnomeui	Includes toolkit extensions to the GTK+ widget set for creating dialog boxes and message boxes, menu bars, toolbars, status lines, and so on. It also includes icons for use in dialog boxes, menu entries, and buttons, and it provides the Gnome canvas for the easy creation of complex interfaces, such as address books, calendar applications, and spreadsheets. This is a toolkit-dependent library currently using the GTK+ Toolkit.
libgnorba	A library for using the ORBit CORBA implementation with Gnome.
libzvt	A library containing a terminal widget.
libart_lgpl	Contains graphic functions used for GnomeCanvas.

Libgnome and **libgnomeui** are the two main libraries needed for any Gnome applications. **Libgnome** is a set of functions designed to be independent of any particular GUI toolkit. These functions could be used in any kind of program, whether it be one with just a command line interface or even no interface. These functions are independent of any particular GUI toolkit. The **libgnomeui** library contains functions that provide GUI interface operations. These are tied to a particular GUI toolset, such as the GTK. It is possible to create a **libgnomeui** library that is tied to a different GUI toolset.

The **libgnome** library provides many utility routines related to the Gnome desktop environment. Among the capabilities provided are **config** file support for applications to store persistent data, support for metadata (data attached to file objects, like the icon that will display for a particular file type), and support for loading help documents into the Gnome help browser. An interface is also provided so Gnome applications can talk to the Gnome session manager. Finally, there are routines to configure how different mime types are handled by Gnome and the Gnome file manager. Some of these functions are listed in Table 4.

Table 4: Gnome: Library (libgnome)

Gnome Library Function Category	**Description**
gnome-config	Provides simple access to configuration values.
gnome-defs	Contains Gnome definitions for C++ linking.
gnome-exec	Permits execution of programs from Gnome applications.
gnome-help	Contains routines for displaying help.
gnome-history	Keeps track of recently used documents.
gnome-i18n	Provides support for localization and internationalization.
gnome-mime-info	Contains routines to get information bound to a MIME type.
gnome-paper	Contains paper dimensions and printing unit conversions.
gnome-popt	Contains the command line argument parser.
gnome-regex	Contains the regular expression cache implementation.
gnome-sound	Includes sound-playing routines for Gnome applications.
gnome-triggers	Contains a hierarchical signal mechanism for application events.
gnome-url	Permits launching viewers for documents based on their URL.
gnome_lib	Initializes **libgnome** library.

The **libgnomeui** library contains the functions and structures you need to create Gnome user interfaces for your applications, and these functions are tied to the GTK+ Toolkit. This library contains toolkit extensions to the GTK+ widget set, and programmers can easily create dialog boxes and message boxes, as well as menu bars, toolbars, and status lines. An extensive array of stock icons is provided for programmers to use in dialog boxes, menu entries, and buttons, and because all Gnome applications will use **libgnomeui** to create these common GUI elements, visual consistency is guaranteed. Similar in many ways to the Tk canvas, the Gnome canvas provides a framework for creating address books, calendar applications, and spreadsheets.

GUI applications require extensive use of images to create a friendly and comfortable user interface. Traditionally, it has been difficult to load all of the common graphic file formats into X11 applications. The **libgdk_imlib** library addresses this issue by providing convenient and powerful functions to load multiple file formats (JPEG, GIF, TIFF, PNG, XPM, PPM, PGM, PBM, and BMP). These files are converted to an internal 24-bit RGB representation, and utility functions exist to scale as well as render from 24-bit RGB to a variety of other color depths (with dithering if desired). Input image files are cached internally by **libgdk_imlib** to improve performance in applications that repeatedly use images.

The **libgtk** library is the GTK Toolkit library. It is a professional-quality widget set that in many ways is superior to other widget sets. Gnome applications are written entirely using **libgtk** for all GUI elements (buttons, menus, scroll bars, and so on). The **libgnorba** library provides support for CORBA operations, such as obtaining references to objects and requesting new instances of objects. **Libzvt** is a simple library containing a terminal widget. **Libart_lgpl** holds graphic functions that can be used with the GnomeCanvas widget.

GTK+

GTK+ consists of an extensive set of functions for widgets of various types, such as menus, buttons, and windows. It also supports bindings that associate GUI events, such as mouse clicks, with objects, such as buttons. Check the online documentation for the GTK API at the **www.gtk.org** and **developer.gnome.org** Web sites. The documentation includes a comprehensive listing of all GTK functions, as well as a detailed tutorial on GTK programming. It is highly recommended that you make use of this documentation—due to size constraints, this book can only present brief introductions and list several of the common GTK functions. Also, check the GTK header files for a detailed declaration of different functions and structures, including their arguments and return values.

Several basic functions and components are needed in any GTK program. You first need to include at least the **gtk.h** header file. Other GTK header files may be required, depending on the widgets and functions you are using. You then have to define pointers to the widgets you intend to define and use. Then you have to initialize the GTK library with the **gtk_init** function. Once that's done, you can define your widgets using GTK functions and assign their addresses to the pointers defined earlier. Then, you can use GTK functions to specify actions and attributes for the widgets, such as displaying them. For example, a close box event (**delete_event**) is connected to the window and the **gtk_main_quit** function so that when a user clicks the Close box of the window, the program ends. Finally, you use the **gtk_main** function to run the widgets.

The following **base.c** program defines a simple GTK program that displays a simple window:

```
#include <gtk/gtk.h>
int main( int argc, char *argv[] )
        {
        GtkWidget *window1;

        gtk_init (&argc, &argv);

        window1 = gtk_window_new (GTK_WINDOW_TOPLEVEL);


        gtk_widget_show (window1);

        gtk_main ();

        return(0);
        }
```

The **gtk.h** header file includes GTK variable, macro, and function definitions. **window1** is defined as a pointer to a structure named **GtkWidget**. The actual structure pointed to will later be determined by the function used to create a given structure. The **gtk_init** function creates initial settings, such as the default visual and color map, and it then calls the **gdk_init** function to initialize the GTK library and check for GTK arguments. The **gtk_window_new** function creates a new window structure, returning its address, which is then assigned to the window pointer. The window is now pointing to the GTK window structure. The GTK_WINDOW_TOP_LEVEL argument will place the window under the control of the window managers, using its window manager's defaults for displaying a window. The **gtk_widget_show** function then displays the window—notice that the window pointer is used as the argument to this function. Finally, the **gtk_main** function starts the interactive process, waiting for events to occur, such as button selections and mouse clicks.

You compile a GTK+ program using the gcc compiler and the GTK+ libraries. To specify the GTK+ libraries on the command line, you use the **gtk-config** command. This command determines the compiler options needed to compile a GTK+ program.

```
 gtk-config --cflags --libs
```

gtk-config is a command that needs to be executed on the command line. To do this, you surround it and its arguments with back quotes. Back quotes are shell operators that are used to execute an enclosed command on the command line and place its returned values in the same place on that line. You can think of this operation as functioning somewhat like a macro, substituting returned values for the command executed. In this case, the **gtk-config** command with the **cflags** and **libs** arguments will place the compiler GTK flags and libraries you need on the command line for the **gcc** command. The **gcc** command is then executed with those flags and libraries:

```
gcc hello.c -o hello `gtk-config --cflags --libs`
```

The libraries usually used are listed in Table 5.

Table 5: Commonly: Used GTK Libraries

Library	**Description**
GTK (**-lgtk**)	GTK widget library
GDK (**-lgdk**)	Xlib wrapper
gmodule (**-lgmodule**)	Runtime extensions
Glib (**-lglib**)	GTK is built on top of Glib and always requires it
Xlib (**-lX11**)	Used by GDK
Xext (**-lXext**)	Shared memory pixmaps and other X extensions
math (**-lm**)	Math library

The program language types used in GTK+ programming can be categorized into fundamental, built-in, and object types. The fundamental types are basic types, such as standard C program types and the base class types for GTK+, like **GTK_TYPE_OBJECT**. The fundamental types are automatically defined by **gtk_init**. The built-in types include some basic enumerations, flags, and structures like **GdkWindow**—these are types that GTK+ need not understand to use. Object types consist of registered **GtkObject** types.

Signal and Events

Gnome programming works like other GUI programming—it is event oriented. In event-driven programs, you first define the objects that the user can operate on, and then you start the interaction function that continually checks for certain events, such as mouse clicks and menu selections. When such an event is detected, it is passed to its appropriate function for handling. For example, if a user clicks on an OK button, the mouse click is detected and control is passed to a function set up to handle a click on an OK button. When the function is finished, it returns control back to the interaction program.

GTK adds a further level of sophistication. When events occur on a certain widget, the widget will emit a signal that is then used to execute a function associated both with that signal and that object. For example, when you click on a Close button, the Close button widget detects the mouse-click event and emits a "clicked" signal. The signal is detected and its associated function is executed.

You can also, if you wish, associate an event directly with a function. For this to work, the programmer has to connect a signal on a given object with a particular function. Functions associated with a particular signal are commonly referred to as "handlers" or "callbacks." When a signal is emitted, its handlers or callbacks are invoked. This process is referred to as "emission." Note that the signals referred to here are in no way like the signals used in Unix systems.

To associate a particular event with the function you want executed for a given signal, you use either the **gtk_signal_connect** or **gtk_signal_connect_object** functions. When the signal is detected, its associated function is automatically executed. The **gtk_signal_connect** function is used for calling functions to which you may be passing arguments, and either **gtk_signal_connect** or **gtk_signal_connect_object** is used for calling functions that require no arguments. In the following **gtk_signal_connect** syntax statement, the **object** is the **GtkObject** you defined, such as a button. The **name** is the name of the signal, such as a mouse click; **func** is the function you want executed whenever an event for this object occurs; and **func_data** are any arguments being passed to that function.

```
gint gtk_signal_connect( GtkObject *object, gchar *name,
           GtkSignalFunc func, gpointer func_data );
```

When a signal is detected for the specified object, its associated callback function is called and executed, as shown in this syntax statement:

```
void callback_func( GtkWidget *widget, gpointer callback_data );
```

Therefore, to associate a click on a button with the **hello** function, you would use the following **gtk_signal_connect** statement:

```
gtk_signal_connect (GTK_OBJECT (mybutton), "clicked",
           GTK_SIGNAL_FUNC (hello), NULL);
```

The object is **mybutton**, **clicked** is the click signal, and **hello** is a function the programmer wrote to be executed when this signal is detected. GTK_OBJECT and GTK_SIGNAL_FUNC are macros that perform type checking and casting to make sure the objects are passed with the appropriate types.

Certain objects have signals that can be associated with them. For example, the button object can be associated with a **clicked** signal or an **enter** signal. The **clicked** signal occurs when a user presses down and then releases the mouse button, whereas an **enter** signal occurs when the user moves the mouse pointer over the button object. The button signals are the following:

- **pressed**   Mouse button pressed down when pointer positioned on the button
- **released**   Mouse button released when pointer positioned on the button
- **clicked**   Mouse button pressed down and released when pointer positioned on the button
- **enter**   Mouse pointer is moved onto the button
- **leave**   Mouse pointer is moved out of the button

You can also use the signal connection functions to connect events directly to an object and function, instead of using signals. Events are messages transmitted by the X11 server to indicate occurrences like mouse clicks and menu selections. In the **gtk_signal_connect** function, you use the name of the event instead of the signal. Callback functions for events include an added argument for the event. The type for this parameter can be **GdkEvent** or one of several other event types. These are listed in Table 6.

```
void callback_func( GtkWidget *widget, GdkEvent *event,
          gpointer callback_data );
```

Table 6: Commonly: Used GTK Events

Event Type	**GtkWidget Signal**
GDK_DELETE	**"delete_event"**
GDK_DESTROY	**"destroy_event"**
GDK_EXPOSE	**"expose_event"**
GDK_MOTION_NOTIFY	**"motion_notify_event"**
GDK_BUTTON_PRESS	**"button_press_event"**
GDK_2BUTTON_PRESS	**"button_press_event"**
GDK_3BUTTON_PRESS	**"button_press_event"**
GDK_BUTTON_RELEASE	**"button_release_event"**
GDK_KEY_PRESS	**"key_press_event"**
GDK_KEY_RELEASE	**"key_release_event"**
GDK_FOCUS_CHANGE	**"focus_in_event", "focus_out_event"**
GDK_SELECTION_CLEAR	**"selection_clear_event"**
GDK_SELECTION_REQUEST	**"selection_request_event"**

For example, to associate a **button_press_event** with an OK button, you would use **"button_press_event"** as the signal name. The following example associates a **button_press_event** event on a button with the **button_press_callback** function:

```
gtk_signal_connect( GTK_OBJECT(button), "button_press_event",
         GTK_SIGNAL_FUNC(button_press_callback), NULL);
```

The callback function used for the signal connection, in this case **button_press_callback**, would have the event type **GdkEventButton** for its event argument.

```
static gint button_press_callback( GtkWidget *widget,
     GdkEventButton *event, gpointer data );
```

The following example associates a click on a window Close box with the **close-win** function. The object is **mywindow**, **delete_event** is the Close-box event, and **close-win** is a function the programmer wrote with code to be executed when this event occurs. When a user clicks on the window's Close box, the **close-win** function is called.

```
gtk_signal_connect (GTK_OBJECT (mywindow), "delete_event",
      GTK_SIGNAL_FUNC (close-win), NULL);
```

Signals are stored in a global table. You can create your own signals with the **gtk_signal_new** function, and then use **gtk_signal_emit** to have an object emit a signal. **gtk_signal_new** will return an identifier for the new signal. You can use this with **gtk_signal_emit** to have your object emit that signal.

Gnome Functions

Gnome programs build on GTK+ programs providing Gnome functions to let you more easily create Gnome interfaces that are consistent with the style for the Gnome desktop. To create a simple GTK program, you begin with GTK object definitions for your Gnome widgets and then use Gnome functions to initialize your program and define your widgets. GTK functions such as **gtk_signal_connect** are used to associate GUI events with objects, whereas Gnome functions such as **gnome_app_create_menus** create menus. In a Gnome program you need to include an initialization function called **gnome_init**, which you place at the very beginning. To create a primary window for your application, you use **gnome_app_new**.

The following example shows the use of the **gnome_init** and the **gnome_app_new** functions. The **gnome_init** function takes as its arguments any initial arguments that the user must enter when the program starts, as well as an application ID and version number. The user's initial arguments are managed by the **argc** and **argv** special variables. **gnome_app_new** takes as its arguments the title you want displayed in the application window and the name of the application object. It returns the address of the new object which, in this example, is assigned to the **app** pointer. **app** is a pointer to an object of type **GtkWidget**.

```
GtkWidget *app;

gnome_init ("", "0.1", argc, argv);
app = gnome_app_new ("Hello-World", "Hello App");
```

Other operations, such as displaying widgets and starting the interactive interface, are handled by GTK functions. **gtk_widget_show_all** will display a widget and any other widgets it contains. **gtk_main** will start the interactive operations, detecting GUI events such as mouse clicks and key presses and executing their associated functions.

```
gtk_widget_show_all(app);
gtk_main ();
```

Compiling Gnome Programs

Given the extensive number of libraries involved in creating Gnome applications, the compiler command with all its listed libraries and flags can be very complex to construct. For this reason, Gnome provides the **gnome-config** script. You place a call to this script as an argument to the compiler operation instead of manually listing Gnome libraries and flags. **gnome-config** takes two options, **-cflags** and **-libs**. The **-cflags** option will generate all the flags you need, and the **-libs** option generates the list of necessary Gnome libraries. You do need to specify the libraries you want to use, such as **gnomeui** and **gnome**, as shown here:

```
gnome-config --cflags --libs gnome gnomeui
```

For the compiler operation, you would place the **gnome-config** operation in back quotes to execute it:

```
gcc myprog.c -o myprog 'gnome-config --cflags --libs gnome gnomeui'
```

To simplify matters, you can place this operation in a **Makefile**. In a **Makefile**, the compiling is performed separately from the linking. For compiling, you would use a **gnome-config** script with the **-cflags** option, and for linking you would use the **-libs** option. In the following example, the CFLAGS and LDFLAGS macros are used to hold the compiling and linking results, respectively. Notice the use of back quotes in the code.

```
makefile
 CFLAGS='gnome-config --cflags gnome gnomeui'
 LDFLAGS='gnome-config --libs gnome gnomeui'

 all: bookrec

 bookrec: file.o calc.o
       cc $(LDFLAGS) main.o -o bookrec
 main.o: main.c
        cc $(CFLAGS) main.c
 file.o: file.c file.h
        cc $(CFLAGS) file.c
```

Gnome Program Example

The **hello1.c** program is a simple Gnome application in the "Hello World" tradition. The program creates a simple window with a button that displays a message on the standard output of your terminal window. When the user clicks the Close box (**delete_event**), the window closes (see Figure 1).

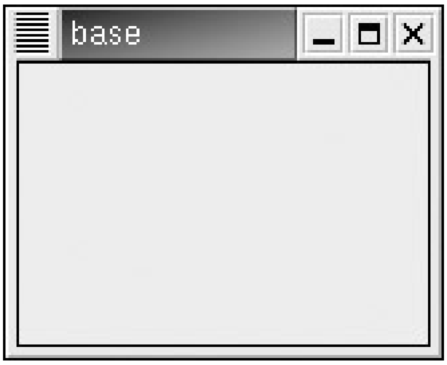


Figure 1: Gnome window

Gnome functions begin with the term "**gnome**", whereas GTK functions begin with "**gtk**". Notice that the initialization function is a Gnome function, **gnome_init**. As explained earlier, Gnome programs are event-driven: you first define your objects, such as windows, then set their attributes, and then bind signals from events such as mouse clicks to objects like windows and to functions that process these events. Such functions are often referred to as callback functions.

To compile this program, you can use the following compile command in a Gnome terminal window. Then, just enter **hello1** to run it. The **-o** option specifies the name of the program, in this case **hello**. Be sure to use back quotes for the **gnome-config** segment.

```
gcc hello1.c -o hello1 'gnome-config --cflags --libs gnome gnomeui'
```

You would use the following steps to create the **hello** program listed in **hello.c**.

- Define two callback functions: **hellomessage** and **closeprog**. **hellomessage** just outputs a simple text, "Hello World". **closeprog** invokes the **gtk_main_quit** function to end the program.
- In the main function, define two **GtkWidget** pointers: **app** and **mybutton**. **app** should be a pointer to the main application window and **mybutton** to a simple button object.
- Create a **gnome_init** function to initialize the Gnome interface.
- Create a button object using the **gtk_button_new_with_label** function, and assign its address to the **mybutton** pointer, as shown in the following code line. The button will be displayed with the label "Click Me".

  ```
  mybutton = gtk_button_new_with_label("Click Me");
  ```

- Create an application window widget using the **gnome_app_new** function, and assign its address to the **app** pointer.
- Use **gnome_app_set_contents** to place the button in the application window.
- Use **gtk_signal_connect** to connect the application with a **delete_event** signal, which occurs when the user clicks the Close box. Set this to execute the **closeprog** function, which should use **gtk_main_quit** to end the program.
- Use **gtk_signal_connect** to connect the button to the mouse click event (**clicked**), and set this to execute the **hello** function. Whenever the user clicks the button, "Hello World" should be displayed on the standard output.
- Use the **gtk_widget_show_all** function to display the application window and the button it now contains.
- Use **gtk_main** starts the interactive interface.

The contents of the **hello1.c** program are shown here.

hello.c

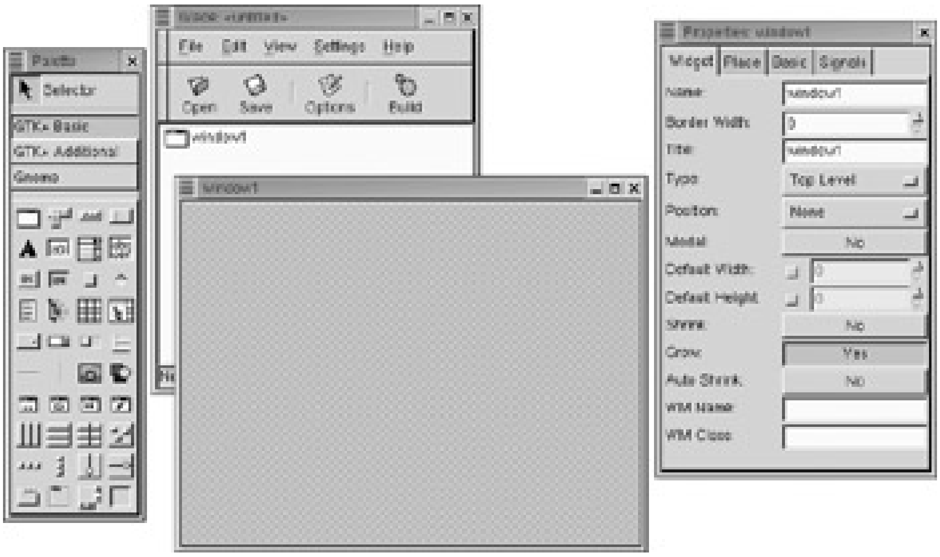
```
#include <gnome.h>

 void hellomessage( GtkWidget *widget, gpointer data )
           {
            g_print ("Hello World\n");
            }

 gint closeprog ( GtkWidget *widget, GdkEvent *event,
 gpointer data )
            {
            gtk_main_quit();
            }
```

```
int main( int argc, char *argv[] )
           {
           GtkWidget *app;
           GtkWidget *mybutton;

           gnome_init ("", "0.1", argc, argv);

           mybutton = gtk_button_new_with_label("Click Me");
           app = gnome_app_new ("Hello-World", "Hello App");
           gnome_app_set_contents (GNOME_APP (app), mybutton);
           gtk_signal_connect (GTK_OBJECT (app), "delete_event",
           GTK_SIGNAL_FUNC (closeprog),NULL);

           gtk_signal_connect (GTK_OBJECT (mybutton), "clicked",
           GTK_SIGNAL_FUNC (hellomessage), NULL);
           gtk_widget_show_all(app);
           gtk_main ();

           return(0);
           }
```

Glade

Instead of coding complex and detailed statements for all your Gnome widgets, you can use Glade to automatically generate them. Glade provides a graphical user interface for creating Gnome widgets, combining them into a GUI interface for your application. With Glade, creating a Gnome interface is as simple as selecting and placing widgets onto windows. For each widget you can specify certain properties such as their size, color, and the signals they use. See **glade.gnome.org** for more information.

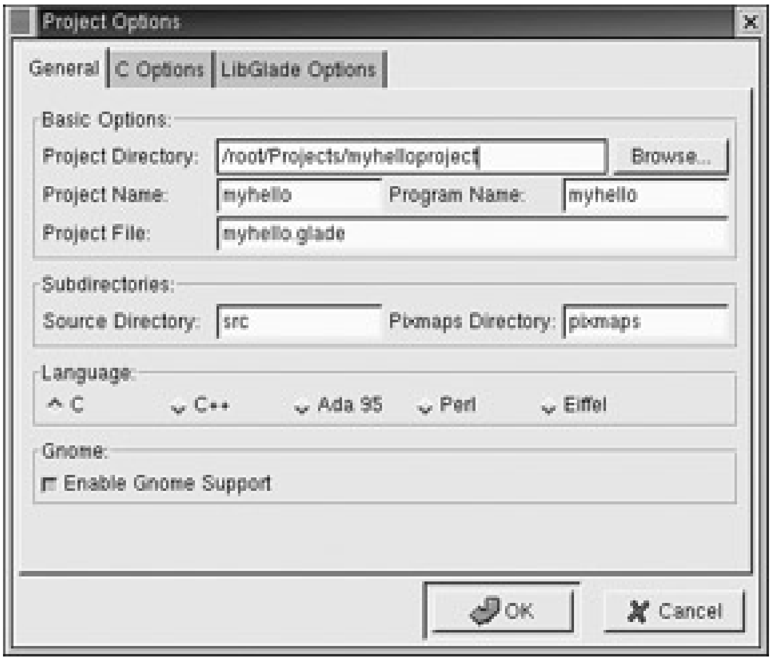
You can start Glade by selecting its entry in the Gnome Applications menu. Be sure that you have already installed the Gnome development packages, including Glade. Glade will initially display three separate windows, the main Glade window, the Palette window, and the Property Editor (labeled Properties). To create your interface you will be selecting Gnome widgets in the Palette window. When you select a window on the Palette, a window will automatically be generated on your desktop. You can then select different widgets such as buttons and menus, and place them on the window, building your interface. When you create a window, it is listed in the main Glade window. As you create more windows for your application interface, icons for them will be displayed in the Glade window. Figure 2 shows Glade with a window labeled window1 that has been created. From the Glade window you can also open previous projects. When you are finished with Glade, select Exit from the File menu.



Name: window1
Border Width: 0
Title: window1
Type: Top Level
Position: None
Modal: No
Default Width: 0
Default Height: 0
Shrink: No
Grow: Yes
Auto Shrink: No
WM Name:
WM Class:

Figure 2: Glade

To start using Glade, you first create a project. Select New Project from the File menu or click on the New icon in the toolbar. The Project Options window will let you select the directory you want your project placed in (see Figure 3). By default, projects are placed in a directory called Projects in your home directory. Each project has its own subdirectory which is given the default name numbered project1. You can specify your own name for the Project directory, as well as the subdirectories for the particular projects. A Glade project file will have the extension **.glade**, in this directory. The source code files for the project are placed in a subdirectory named **src**. Glade will generate a **main.c** and an **interface.c** source code file for the widgets you create.



Basic Options:
Project Directory: /root/Projects/myhelloproject
Project Name: myhello
Program Name: myhello
Project File: myhello.glade

Subdirectories:
Source Directory: src
Pixmaps Directory: pixmaps

Language:
● C ○ C++ ○ Ada 95 ○ Perl ○ Eiffel

Gnome:
□ Enable Gnome Support

Figure 3: Glade project options

The Palette is the key component in Glade. Here you will find all the widgets you can use to create your interface. The Palette displays three different kinds of widgets: GTK+ Basic, GTK+ Additional, and Gnome. There are buttons for each at the top of the Palette window, and clicking on one will display the widgets available for each category. GTK+ Basic, as the name suggests, provides basic components such as buttons, windows, and menus. GTK+ Additional provides more sophisticated, though less commonly used, GTK+ widgets. Gnome lists the Gnome widgets. These conform to the functions in the Gnome libraries. Of particular note is the Gnome application window, which provides a Gnome application window with basic menus and toolbar already installed. Figure 4 shows the Glade Palette displaying the GTK+ widgets.

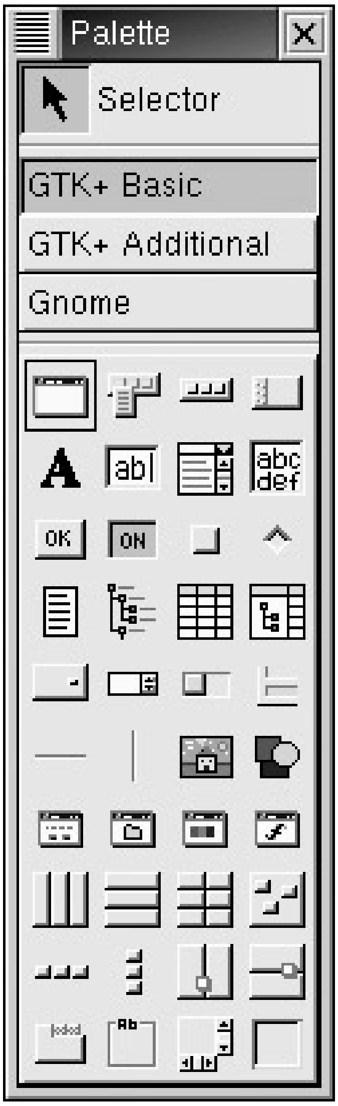


Figure 4: Glade Palette

When creating a Gnome interface, bear in mind that Gnome uses a container method for holding and placing its widgets. Each widget that you place in a window, cannot be placed directly in the window, but, instead, must be placed in a container. In fact, the key to designing a Gnome interface is to first select the appropriate container for the widget you want to add. For example, to place a button in a Gnome window, you first place a container for it on the window and then place the button in that container. Containers come in a variety of combinations. You can have several containers stacked on top of each other in rows (horizontal boxes), or set beside each other in columns (vertical boxes). Containers can be organized into tables, or you can create a container that supports fixed positions. When you place a container on a window, you will be asked to select the number of containers you want. For example, for containers that fill up rows the length of a window, you would select horizontal box container. When you place it on the window, you will be prompted to enter the number of rows you want. The different types of containers are displayed at the bottom of the GTK+ Basic Palette window, showing the outline formats for each. Figure 5 shows the containers currently available.

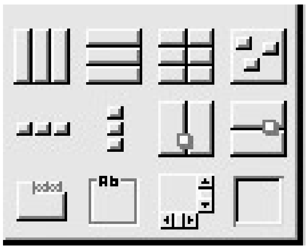


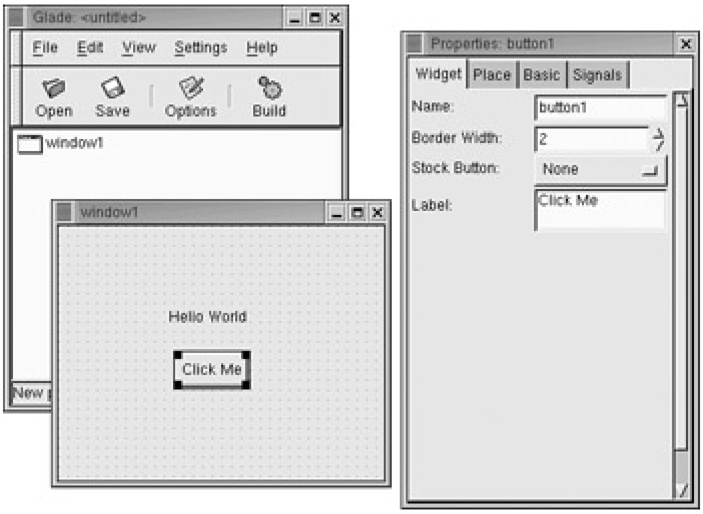
Figure 5: Glade GTK+ containers

Keep in mind that, unless you select the Fixed Position container, the widget will automatically expand to fill the frame. For example, a button placed in a row container will fill the entire length of the window. This is helpful if you are selecting the kind of widget that would fit well into that container. For example, if you want to add a menu and a toolbar to a blank window, you would first select and place the horizontal boxes container and select three rows. Initially, all three rows will be evenly sized. Then you would click on the menu widget and place it on the first row. The row will contract to the size of the menu, and the same for the toolbar placed on the second row.

You can create complex combinations by placing one container inside another. For example, you could first create two row containers, placing a menu in one. Then place two column containers in the remaining row container. In one column, you could place several button-row containers and put buttons in them. The other column could hold a frame for displaying data.

Tip You can delete any widget, including containers, by right-clicking on them and selecting Delete from the pop-up menu.

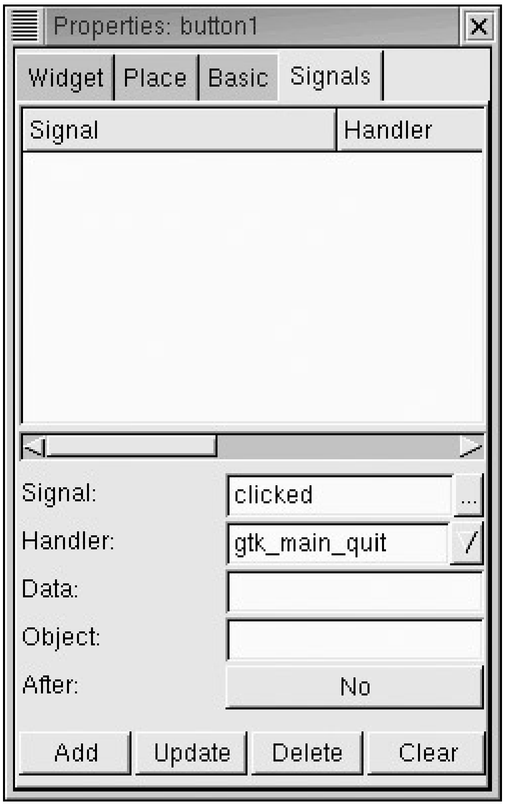
To just place a widget anywhere on a surface, you would use the Fixed Positions container. With this container, you can then select any number of widgets, placing each at different places on the container space. You can move or resize the widget by selecting it to display the anchor points on its corners. Use these to resize it, or use click and drag on the selected widget to move it. In Figure 6, a Fixed Position container has been created and two widgets placed on it—a label with the text "Hello World" and a button with the text "Click Me".



Name: button1
Border Width: 2
Stock Button: None
Label: Click Me

Figure 6: Buttons and labels on Glade

The properties window will display the different properties for a selected widget. The properties window displays several tabbed panels. Here, you can enter the features like the text displayed by a widget—should it display text. Widgets like buttons and menu items will also have a Signals panel where you can select the signals that that widget will respond to. You can have several signals for any given widget. To create a signal, you first select the kind of signal from the pop-up menu, then select the handler function that signal will execute. You can also enter data and objects to be passed. Figure 7 shows the Signal panel for the Click Me button.



Signal: clicked
Handler: gtk_main_quit
Data:
Object:
After: No

Figure 7: Glade Signal Properties panel

Once you have created your interface, you then need to generate the source code for it. Click the Build icon or select Build from the File menu. This will generate **main.c** and **interface.c** files. The **interface.c** file in your project's **src** directory will hold the C code for your widgets. You can then continue to code your program, adding classes and functions your project develops. You can open your Glade project at any time to add new windows, dialogs, and their widgets. In the project directory, you can use Autoconf and configure commands to generate an appropriate **Make** file, and then use the **make** command to create your application.

Gnome App and Menu Widgets

The **GnomeApp** widget is the basic widget for Gnome applications. This widget is the main window holding menus, toolbars, and data. You use the **gnome_app_new** function to create a new **GnomeApp** widget. This function takes as its argument the name of the application.

To add elements such as toolbars, menus, and status bars to the widget, you just use the appropriate function. For example, to add a menu, use **gnome_app_set_menus**, and to add a status bar use **gnome_app_set_statusbar**. To add just a single toolbar, use **gnome_app_set_toolbar**, and to add multiple toolbars use **gnome_app_add_toolbar**.

With the **gnome-app-helper** functions, you can generate menus and toolbars automatically using **GnomeUIInfo** structures. For toolbars and menus, you can create **GnomeUIInfo** structures for them with the appropriate values and then use **gnome_app_create_menus** to create menus and **gnome_app_create_toolbar** to create toolbars.

With the GNOMEUIINFO_ITEM macro, you can add an item to a menu. The GNOMEUIINFO_SEPARATOR macro adds a separator line, and the GNOMEUIINFO_END macro specifies the end of menu. In the following example, **label** is the text of the label, **tooltip** is the tooltip that will be displayed when the pointer moves over that item, and **callback** is the function that is executed when the user clicks that item. You can add another argument for an icon image if you want an icon displayed in the menu item. This is usually a **.xpm** image.

```
GNOMEUIINFO_ITEM(label, tooltip, callback)
```

To specify an accelerator key for a particular item, you just place an underscore before the letter in the **label** for the key you want to use. An accelerator key is an alternative key you can use to access the menu item. This is usually an ALT key. In the following example, the menu item will have an Exit label with the "x" underlined, indicating that you can use an ALT-X key combination to access this item.

```
GNOMEUIINFO_ITEM("E_xit", "Exit the program", exitfunc)
```

The GNOMEUINFO_ITEM macro generates the values to be used in a **GnomeUIInfo** structure. You can assign these values to such a structure. In the following example, a menu is created consisting of an array of **GnomeUIInfo** structures, and **GnomeUIInfo** macros are used to assign values to each **GnomeUIInfo** structure in this array. In this example, a simple File menu is created with two entries, one for Open and one for Exit. A line separator will be displayed between them.

```
GnomeUIInfo file_menu[] = {
            GNOMEUIINFO_ITEM("_Open", "Open a document", openfunc),
            GNOMEUIINFO_SEPARATOR,
            GNOMEUIINFO_ITEM("E_xit", "Exit the program", exitfunc),
            GNOMEUIINFO_END
            };
```

A number of macros are provided for standard menu items, like the Save and Open entries in a File menu. These take as their arguments the function to be executed when the item is selected (**cb**) and any icon image you want displayed for the entry (**data**). Here is the syntax for these macros:

```
GNOMEUIINFO_MENU_OPEN_ITEM(cb, data)
```

The following example creates the same simple File menu as in the previous example, but it uses specialized macros to create each item. Here, the GNOMEUIINFO_MENU_EXIT_ITEM macro creates the Exit entry for the menu:

```
GnomeUIInfo file_menu[] = {
              GNOMEUIINFO_MENU_OPEN_ITEM(openfunc),
              GNOMEUIINFO_SEPARATOR,
              GNOMEUIINFO_MENU_EXIT_ITEM(exitfunc),
              GNOMEUIINFO_END
              };
```

For submenus and for menus added to your menu bar, you use the GNOMEUIINFO_SUBTREE(label, tree) macro, where **tree** is the array of **GnomeUIInfo** structures to be used for that submenu.

The following example assigns the File menu defined earlier and an Edit menu to a menu bar. Again, these are **GnomeUIInfo** structures for which the macros generate values. Notice the use of underscores in the labels to designate ALT keys for accessing the menus.

```
GnomeUIInfo menubar[] = {
              GNOMEUIINFO_SUBTREE("_FILE", file_menu),
              GNOMEUIINFO_SUBTREE("_EDIT", edit_menu),
              GNOMEUIINFO_END
              };
```

For particular menus on a menu bar, you use the menu tree macros. The **tree** argument is the array of **GnomeUIInfo** structures for the menu. For example, the File menu can be added to the menu bar with the following statement, where **tree** is the array of **GnomeUIInfo** structures for the File menu:

```
GNOMEUIINFO_MENU_FILE_TREE (tree)
```

The following example is a rewritten version of the menu bar assignment using specialized macros for the File and Edit menus:

```
GnomeUIInfo menubar[] = {
              GNOMEUIINFO_MENU_FILE_TREE(file_menu, NULL),
              GNOMEUIINFO_MENU_EDIT_TREE(edit_menu, NULL),
              GNOMEUIINFO_END
              };
```

Once you have defined your menus, you can create them using the **gnome_app_create_menus** function. This takes as its arguments the Gnome application structure and the pointer to the **GnomeUIInfo** structures you are using for your menu bar. In the previous example, this pointer was the array name "menubar". Each of the elements making up the **menubar** array, in turn, references a **GnomeUIInfo** array for their menu.

```
gnome_app_create_menus (GNOME_APP (app), menubar);
```

KDE Development: KDevelop

KDE (K Desktop Environment) is organized on a C++ object model with C++ objects containing functions with which you can modify the object. Many of the functions are inherited from higher-level KDE classes, while others are defined for a particular type of object. In a KDE program, you define an object and then use its public and private functions to modify it. For example, you can create a menu object and then use the menu object's functions to add new menu items to it. KDE uses the Qt Toolkit which is developed by Troll Tech (**http://www.troll.no**). It is this toolkit that is actually used to display and manage GUI objects such as buttons and windows. The Qt Toolkit operates much like the GTK+ Toolkit in Gnome.

Because KDE applications are C++ object-oriented programs, they use a set of hierarchical object classes contained in the KDE and Qt libraries. Classes lower in the hierarchy will inherit members (functions) from predefined KDE classes higher in the hierarchy, and you can create your own classes and have them inherit members. KDE uses the Qt Toolkit and currently relies on it directly. Unlike Gnome, which can have its lower-level functions managed by any toolkit, KDE relies solely on the Qt Toolkit. Currently, KDE programming is essentially Qt programming.

KDE and Qt programming rely on an extensive set of classes, each of which usually has a significant number of member functions that manage objects of that class. There are far more than can be listed within the size limitations of this book. For a complete listing of the KDE user interface classes, consult the documentation provided on the KDE developer's site, **developer.kde.org**. This site includes detailed tutorials, and complete reference materials for the KDE API as well as KOM (KDE Object Manager) documentation and Qt reference material. Each class is described in detail, and class type declarations, including their member function declarations and definitions, are given. In addition, consult the KDE and Qt header files. The **.h** files contain a complete listing of the KDE and Qt classes, along with detailed comments describing their member functions.

A widget, like a window or a button, is just an object. You can define a window object using a KDE or Qt window class or a button using a KDE or Qt button class. There are several kinds of classes that you can use, depending on the type of window or button you want. To create a complex widget, such as a window that contains other widgets (perhaps toolbars and menus), you would define the subwidgets as children of the main widget. When you define a toolbar, you specify a particular window object as its parent. A subwidget can, in turn, have its own subwidgets, its own children. For example, a menu bar can have a window as its parent and individual menus as its children.

When you declare a C++ object, you usually include arguments in addition to the class and object name. These arguments are passed to a special function called a *constructor* that is executed when the object is defined, which performs any needed setup or initialization operations for the object.

For widgets, one of these arguments is usually the address of its parent widget. For example, a toolbar will be defined with one of its arguments being the address of a window object that is its parent. If the widget is a top-level object with no parent, the argument is NULL. With a series of simple object definitions, you can easily create a complex widget.

KDE Libraries

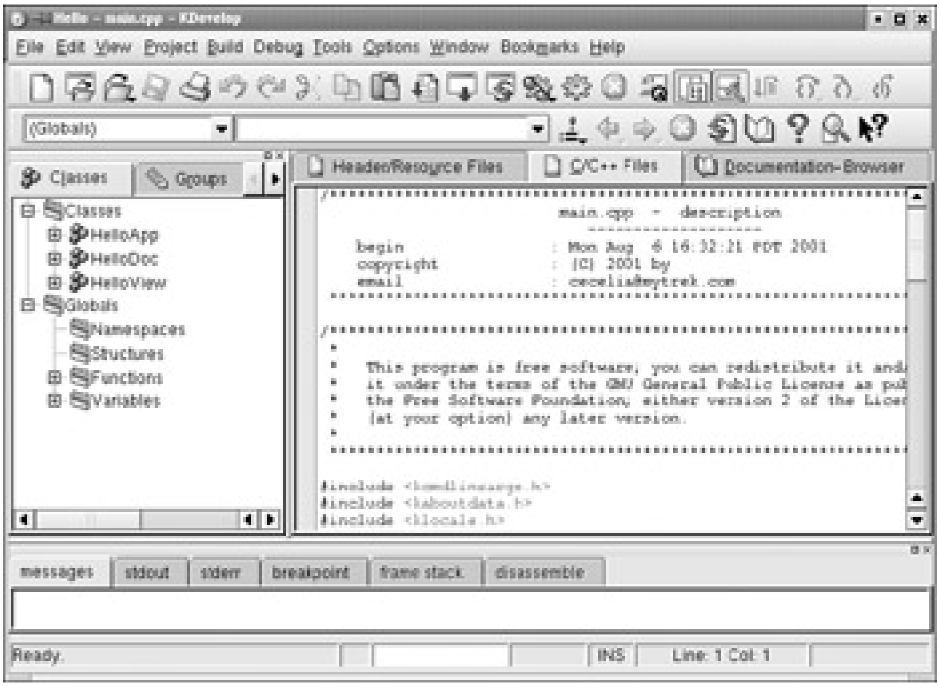
A KDE program is simply a C++ program that uses objects whose classes are defined in the KDE and Qt libraries. You use the g++ compiler on your source code files as you would any other C++ program. g++ is the C++ form of the gcc C compiler. There are several KDE libraries, each with an extensive set of classes. Most programs will need at least the **kdecore** and **kdeui** libraries. **Kdeui** holds the KDE user interface classes for KDE widgets (see Table 7).

Table 7: Common: KDE Kdeui (User Interface)

Widget	**Description**
DialogBase	A base class that provides basic functionality needed by nearly all dialog boxes
KApplet	The KDE Panel Applet class
KButton	The class that provides active raise/lower buttons
KButtonBox	A container widget for buttons
KCursor	A Qt QCursor wrapper allowing "themed" cursors
KDialog	A dialog box with extended modeless support
KFontChooser	A widget for interactive font selection
KGradientSelector	A gradient selector widget
KMenuBar	A floatable menu bar
KMessageBox	An easy MessageBox dialog box
KNumCheckButton	A different type of Check button
KPopupMenu	A pop-up menu with a title
KProgress	A progress-indicator widget
KSeparator	A standard horizontal or vertical separator
KStatusBar	A KDE status bar widget
KStatusBarItem	An internal class for use in KStatusBar
KStatusBarLabel	An internal class for use in KStatusBar
KTMainWindow	A KDE top-level main window
KToolBar	A floatable toolbar with auto-resize
KToolBarButton	A toolbar button
KToolBarItem	A toolbar item
KTopLevelWidget	An old KDE top-level window

KDevelop

KDevelop is an integrated development environment (IDE) for writing K Desktop programs (see **www.kdevelop.org**). You can access KDevelop from the Development entry in the K Desktop menu. When KDevelop first opens, it displays a window with two toolbars, a status bar, and three subwindows (see Figure 8). The bottom window displays debugging information. The left window will display the different C++ classes and source code files defined for your project. The right window holds three tabbed panes: one for header files, one for your source code files, and the other for KDevelop documentation.

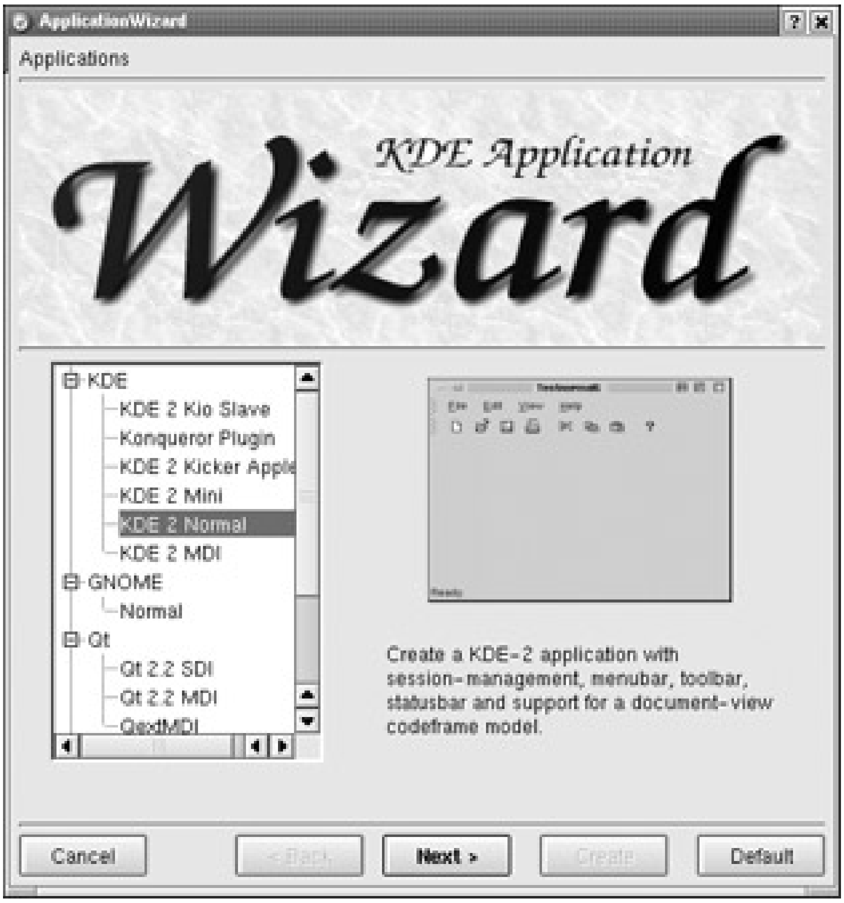


```
/***************************************************************************
                          main.cpp  -  description
                             -------------------
    begin                : Mon Aug  6 16:32:21 PDT 2001
    copyright            : (C) 2001 by
    email                : cecelia@mytrek.com
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and
 *   it under the terms of the GNU General Public License as pu
 *   the Free Software Foundation; either version 2 of the Lice
 *   (at your option) any later version.
 *                                                                         *
 ***************************************************************************/

#include <kcmdlineargs.h>
#include <kaboutdata.h>
#include <klocale.h>
```

Figure 8: KDevelop

Note KDevelop and Qt Designer are not included with the Publisher's Edition. You will need to download them from the Red Hat FTP site.

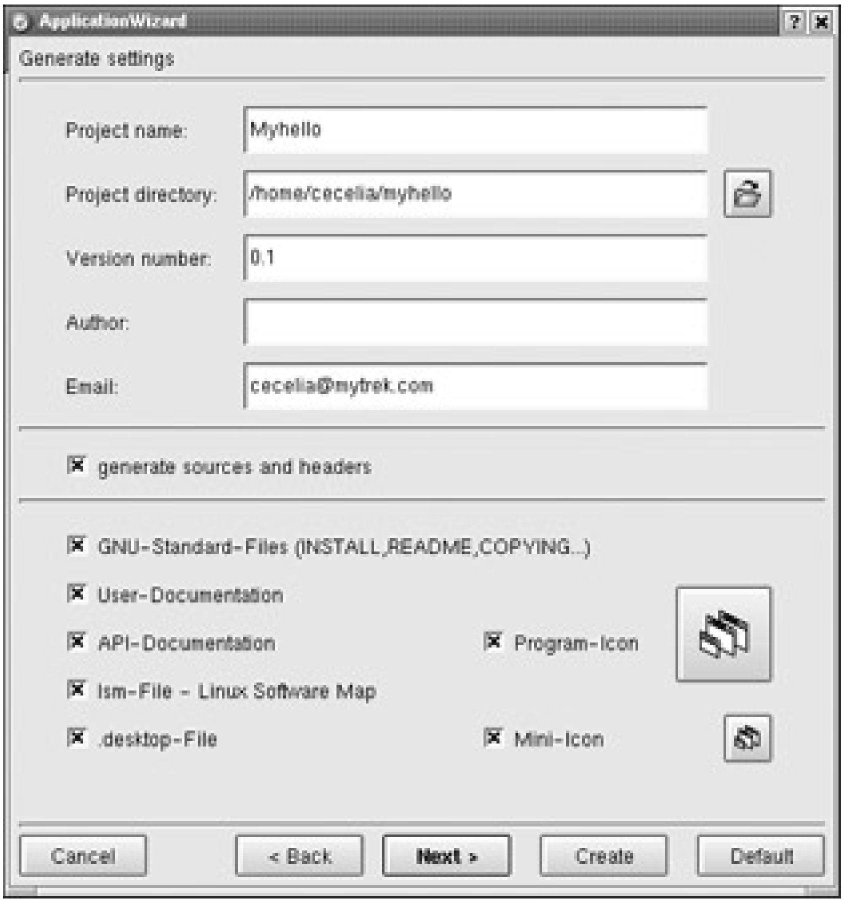
To start a project you select New from the Project menu. This will start up the Application Wizard, shown in Figure 9, that helps you set up the kind of application you want to create. For example, you could create a KDE panel applet, a Konqueror browser plug-in, or a standard KDE Desktop interface.



Create a KDE-2 application with session-management, menubar, toolbar, statusbar and support for a document-view codeframe model.

Figure 9: KDevelop Application Wizard

Click the Next button to move to the next window in the Wizard. Here, you specify the project name and its directory, along with the kind of support files you will need, such as those for documentation (see Figure 10). KDevelop will create a directory with the same name as the projects. Within that directory, it will create a further subdirectory with the same name, which will hold the source code files. The source code files for the myhello project would be in the **myhello/myhello** directory.



ApplicationWizard

Generate settings

Project name: Myhello

Project directory: /home/cecelia/myhello

Version number: 0.1

Author:

Email: cecelia@mytrek.com

☒ generate sources and headers

☒ GNU-Standard-Files (INSTALL,README,COPYING...)

☒ User-Documentation

☒ API-Documentation

☒ Program-Icon

☒ lsm-File - Linux Software Map

☒ .desktop-File

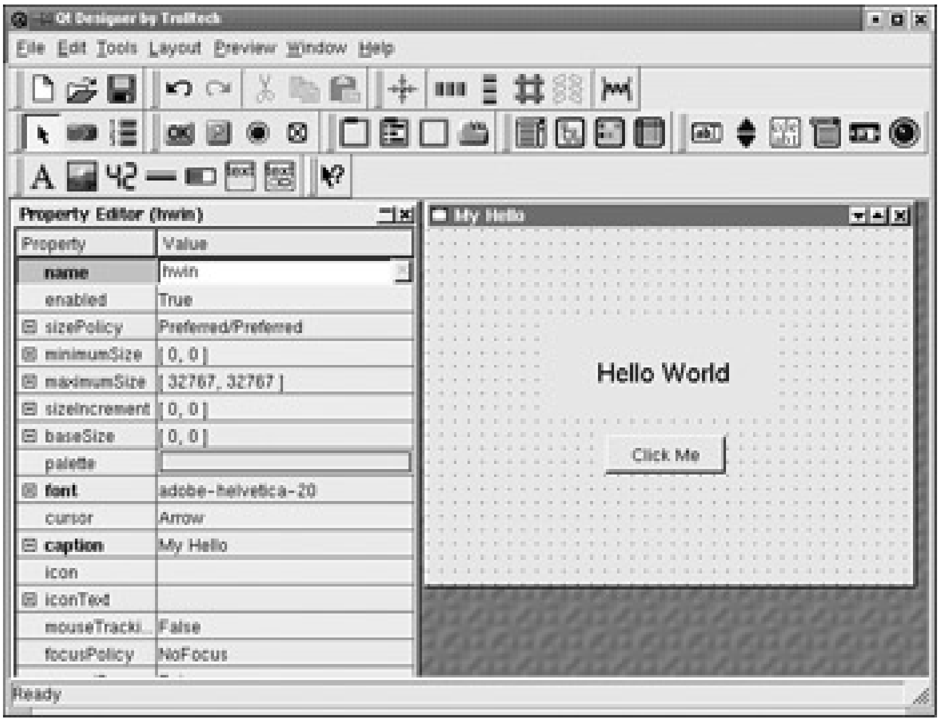
☒ Mini-Icon

Figure 10: KDevelop project specifications

On the next screen you can choose to use the CVS versioning system, to help keep track of changes and enable many users to work on the same project. Then, templates for standard headings in the source and header files are presented. On the Process screen, you click the Create button to have KDevelop generate any needed files and create an initial **main.c** source code file with required K Desktop functions and classes defined, as well as links to the needed KDE libraries. Click the Exit key when finished.

For a standard KDE application, KDevelop will automatically generate the code for an application window, standard menus including File, Edit, and Help menus, a standard icon bar, and a status bar. This initial code will be placed in the source code file of the same name—for example, **myhello.cpp**. You can then add in your own classes with their own function calls and have them invoked from entries you can add to these menus or to the toolbar.

You could code your program as you would any C++ program. However, KDevelop also includes the Qt Designer that you can use to automatically create your KDE windows and widgets. Once you have created your project, you can then start up Qt Designer by selecting the Dialog Editor entry in the View window. This opens a separate Qt Designer window, with toolbars for different KDE widgets (see Figure 11). A subwindow in the main pane is the Property Editor, which will display properties for any selected widget you have created.



Property	Value
name	hwin
enabled	True
⊟ sizePolicy	Preferred/Preferred
⊞ minimumSize	[0, 0]
⊞ maximumSize	[32767, 32767]
⊟ sizeIncrement	[0, 0]
⊟ baseSize	[0, 0]
palette	
⊞ font	adobe-helvetica-20
cursor	Arrow
⊟ caption	My Hello
icon	
⊞ iconText	
mouseTracki...	False
focusPolicy	NoFocus

Figure 11: Qt Designer

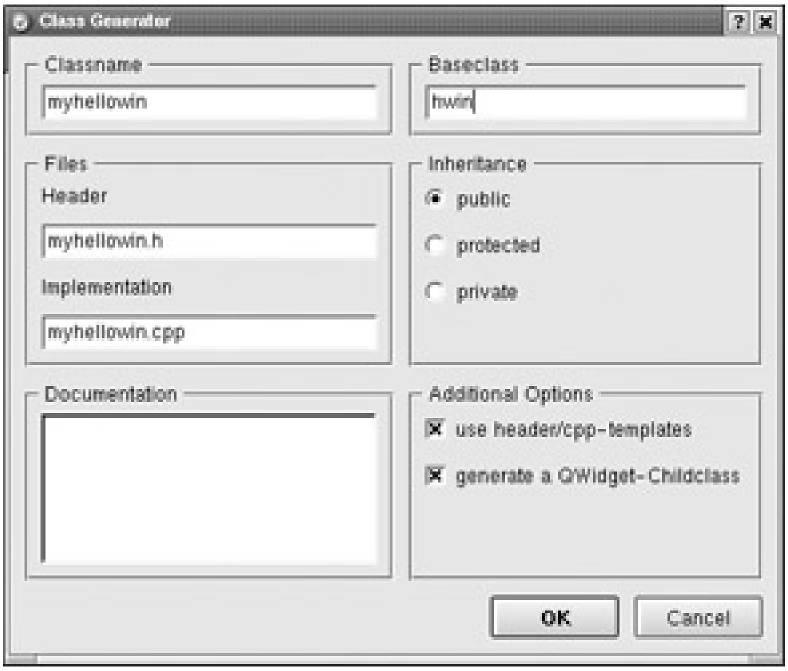
To create a window, either click on the New icon in the toolbar or select New from the File menu. A window will be displayed on the right side of the main pane. To the left will be the window's Property Editor. Here, you can specify the class name of the window, along with its caption, size, and color, and so on. In Figure 11, the class name is **hwin** and the caption is My Hello. When you save the window and its widgets, you need to save them in the directory for the source code used for your project.

To add a widget to your window, click on the widget in the toolbar and then click on where you want to place it on the window, and drag to set the size of the widget. The widget will appear. The Property Editor will automatically display the properties for the selected widget. For example, to create a button, click the Button icon and then click in your window. You can then change the text displayed for the button in its Property Editor window. Figure 11 shows both a label and a button widget added to the hwin window. The button has had its display text changed to "Click Me".

Widgets, like buttons, need to be connected to slots for them to have any effect. The slot is the function that will activate when a specified event occurs on that button. To set up an event/slot connection for a widget, you need to select the Connect Signal/Slots entry in the Tools menu, or click on the Connect Signal/Slots button in the second toolbar. Then, click on the widget. This opens a window where you can select the event and the slot to be associated with it.

Once you have created your windows and widgets with the Qt Designer, you then need to integrate them into your project as source code. When you exit the Qt Designer, KDevelop will set up the necessary instructions in your project's **Makefile** to do this. Qt Designer actually saves its code in special **.ui** files that need to be converted to C++ code to be used. KDevelop sets up this process for you. However, you still need to integrate the KDE windows and widgets into your program. When you created your main window in Qt Designer, you gave it a class name. The example in Figure 11 used **hwin** as the class name for the main window. You have to integrate this class name into the overall class structure of your KDE C++ program. You do this by creating a new class in your program that will inherit from the class you created with the Qt Designer. This way, all those windows and widget classes that contain the code to create and manage those widgets are inherited automatically into your program and can be referenced by the new class you created.

To create a new class, select New Class from the Projects menu. This opens a window where you can specify the name of the new class and any class it inherits from. In Figure 12, a new class called **myhellowin** is created that will inherit from the **hwin** class, its base class. Through the **myhellowin** class, the program can access the window and widgets created for the **hwin** class. Click "Generate a QWidget Childclass" since the classes are derived from the parent **QWidget** class. Whenever you create a new class, KDevelop generates a corresponding header file for it (**.h**). If the class inherits from another class, KDevelop will generate an include statement to include the header file for that class. For example, when the **myhellowin** class is created, a **myhellowin.h** file is generated for it. Since that class inherits from the **hwin** class, an include statement in inserted in **myhellowin.h** that will include the **hwin.h** file. The **hwin.h** file is automatically generated by KDevelop using the **hwin.ui** file created with Qt Designer.



Classname: myhellowin

Baseclass: hwin

Header: myhellowin.h

Implementation: myhellowin.cpp

Inheritance: ● public ○ protected ○ private

Additional Options: ☒ use header/cpp-templates ☒ generate a QWidget-Childclass

Figure 12: New class in KDevelop

Once you have coded your program, you can build and run it by clicking on the Run icon. You can also elect to just build the project or compile selected files.

Tip You can manually generate source code from a Qt Designer **.ui** file using the uic tool. Redirect the output to the appropriate **.h** file—for example, **uic myhello.ui > myhello.h**.

KDE Applications and Widgets

To create a KDE application, you simply create an object of type **KApplication** in your program before you define any other KDE objects. The class declaration for **KApplication** is contained in the **kapp.h** file. The definition of a **KApplication** object takes as its arguments

argc and **argv**. These operate as they do in C programs, referencing any command line arguments the user enters. The following example defines an application object called **myapp**:

```
KApplication myapp( argc, argv );
```

Declarations for different kinds of KDE and Qt objects are located in separate header files. Whenever you define an object of a particular type, be sure to include the header file that has its class declaration. For example, to create a main application window, you use **KTMainWindow** class, and you need to include the **ktmainwindow.h** header file.

```
#include <ktmainwindow.h>
```

The header files are also extremely helpful as a reference source. They list all the member functions for a particular class and include detailed comments describing each function and its use. The header files will be located in the KDE include directory. Currently for Red Hat, this is the standard include directory, **/usr/include**. On OpenLinux and other distributions, it may be the special KDE directory, such as **/opt/kde/include**.

To define a main window for your application, you use the **KTMainWindow** class. The following example defines a main window object called **mywin**:

```
KTMainWindow mywin;
```

If you create an application where the main window is the primary interface and you want the application to close when that window closes, you have to make it the main widget for the application. To do this, you use the application object's **setMainWidget** function. The main widget could be any widget you want. Be sure to pass the address of the widget. You do this by preceding it with the address operator, the ampersand - **&**. The following example sets the main widget to be the **mywin** window. The address of the **mywin** widget is passed, **&mywin**.

```
myapp.setMainWidget(&mywin);
```

Tip If you are using a pointer to a widget as in the later program examples, you need only pass the pointer without using the address operator. A pointer already holds the address.

When you define a widget, you will also be defining any of its member functions contained in its class declaration. See the developer.kde.org documentation for a complete description of all KDE class declarations, including their member functions. Many of these member functions are designed to let you change the display features of a widget, such as its color or initial size. For example, to control the display size of the **KTMainWindow** widget, you use its **setGeometry** function, as shown here:

```
mywin.setGeometry(100,100,200,100);
```

You have to explicitly instruct KDE to show any widget that you want displayed. To do this, you use your widget's **show** member function. For example, to have the **mywin** window display, you execute its **show** function as shown here:

```
mywin.show();
```

Once you have defined all your widgets and made any modifications, you can then run the application. You do this with the **KApplication** object's **exec** member function.

```
myapp.exec();
```

When the user closes the application, control returns to the main function, which can then terminate the program. Usually, the **return** statement with the **exec** function will return any errors that **exec** may return.

```
return myapp.exec();
```

The following program creates a simple KDE application that displays a window:

```
#include <kapp.h>
#include <ktmainwindow.h>

int main( int argc, char **argv )
     {
      KApplication myapp( argc, argv );
      KTMainWindow mywin;
      mywin.setGeometry(100,100,200,100);

      myapp.setMainWidget(&mywin);
      mywin.show();
      return myapp.exec();
     }
```

Many systems will have set up the KDEDIR and QTDIR shell variables. KDEDIR would contain the path names for the KDE commands, headers, and libraries, and QTDIR would contain the path names for Qt. To specify the header files for KDE you would use the following:

```
-I/$KDEDIR/include
```

Be sure to include the **$** before KDEDIR. If KDEDIR is not already defined, you can define it yourself, assigning the location (if you know it) of the KDE components:

```
KDEDIR = /opt/kde
```

On Red Hat Linux, the Qt libraries for 6.0 are placed in the **/usr/lib/qt** directory, and KDE libraries are mixed with other libraries in the **/usr/lib** directory. You will not have to specify a KDE library, and for Qt, you specify the **/usr/lib/qt** directory. Caldera OpenLinux, along with other distributions, currently defines the KDE libraries to be in the **/opt/kde** directory.

In the following example, both the KDE and Qt libraries are specified for the **myapp.cpp** KDE program.

```
g++ -I$KDEDIR/include -L$KDEDIR/lib -I$QTDIR/include -L$QTDIR/libs
 –lkdecore –lkdeui –lqt myapp.cpp
```

Signals and Slots

KDE and Qt use signals and slots to allow one widget to communicate with another. Signals and slots are member functions defined in a class that have special capabilities. *Signals* are emitted by an object when it is activated by some event occurring on it. For example, when a user clicks a **button** object, the button will emit a **clicked** signal. This signal can then be picked up by any other object set up to receive it. Such an object will have *slots* that are designated to receive the signal. A slot is just a member function that executes when the object receives a certain signal.

In effect, slots operate like event handlers, and signals can be thought of as events, but KDE and Qt do not operate like standard event-driven GUIs. Instead, the process of event handling is implemented as messages are sent and received by objects. Instead of focusing on the processing of an event when it occurs, objects manage their own event tasks as they occur, whether that be receiving or sending signals. A KDE widget emits a signal when an event occurs on it or when it changes state for some reason. There are several possible signals, among the more common of which are the **activated** and **clicked** signals. So, when an **activated** signal occurs on a menu item widget, the processing function will execute the corresponding function for that item. For example, given a window with a menu that has an Exit item, when a user clicks on an Exit item in the File menu, a function to exit the program should be executed. The Exit item emits a signal that is then received by the main **window** object, which then executes the slot function associated with the Exit item.

The connection between the signal from an emitting object to a slot function in a receiving object is created with the object's **connect** function. The **connect** function sets up a connection between a certain signal in a given object with a specific slot function in another object. Its first argument is the object, the second is the signal, and the last is the callback function. To specify the signal, you use the SIGNAL macro on the signal name with its parameters. For the callback command function, you use the SLOT macro. Using **connect** operations, you can also connect a signal to several slots and connect several signals to just one slot. In the following example, the clicked signal on the **buttonhi** object is connected to the **myhello** slot function in the **mywin** object:

```
connect(buttonhi, SIGNAL(clicked()), mywin, SLOT(myhello()));
```

Classes composed of several widgets, such as an application window, will often have connections from signals from the different widgets to the main widget. **connect** operations are usually placed with the class declaration of the main widget for connecting signals from its subwidgets to itself. In this case, the main widget (object) can then be referenced with the C++ **this** pointer reference, which always references the class being declared, as shown next:

```
connect(buttonhi, SIGNAL(clicked()), this, SLOT(myhello()));
```

Tip Any class that includes slots or signals must also include a special reference named **Q_OBJECT**. This enables the Meta-Object Compiler preprocessor (described next) to set up any signals and slots declared in the class.

Meta-Object Compiler: MOC

Though the code for entering signal and slot functions, as well as that for making the connections, may appear straightforward to the programmer, it actually requires much more complex C++ coding. Signal and slot functions need to be preprocessed by the Meta-Object Compiler (MOC) to generate the C++ code that can implement the signal and slot message-connection process. You then include the output of MOC in your source code file for compiling. This means that you should place the class declarations for any classes that have signals and slots in separate header files. You can then preprocess these header files and include the output in the source code.

You cannot combine the member function definitions with the class declaration. To compile, the class declaration has to first be preprocessed by MOC before it can be combined correctly with the member function definitions. This necessitates placing the class declaration in a separate file from the member functions so that the class declaration can be separately preprocessed by MOC.

To declare a class that contains either signals or slots, you would first declare the class in a header file like **myobj.h**. You do not place the definitions of any of the member functions in the header file, only the class declaration. Note that the class declaration will include declarations of the member functions, structures, and variables. In a separate source code file, you would place the definition of member functions, like **myobj.cpp**. A member function definition is the actual code for the function.

For these definitions to be correctly compiled, you have to include the MOC preprocessed version of its object declaration, not the actual declaration itself. To generate the preprocessed MOC versions, you use the class declaration header file and the **moc** command, like this:

```
moc myobj.h -o myobj.moc
```

In the particular source code files where you are defining member functions for this object, you would include the MOC version of the header file that contains the object declaration, not the header file itself. So you would include **myobj.moc** in the **myobj.cpp** source code file, not **myobj.h**.

However, for any other source code files where you are generating an object of that class—say, with a **new** operation—you just include the header file, not the MOC file. So, for any source code file where you only need the class declaration, you include the header file, such as **myobj.h**.

For example, suppose in the **main.cpp** file a **myobj** object is generated as a variable, whereas in a **myobj.cpp** file there are function definitions for member functions for the **myobj** class. Furthermore, suppose the class definition for **myobj** is in the **myobj.h** header file and the MOC version of **myobj.h** is in the **myobj.moc** file. In the **main.cpp** file, you would include the **myobj.h** file (not **myobj.moc**), but in the **myobj.cpp** file you would include the **myobj.moc** file (not **myobj.h**).

KDE Program Example

hellowin.cpp is a simple program that displays a button in the main window and then will display a message box with "Hello World" when clicked. The **Hellowin** class will have two slots declared: **myhello** (to display the message) and **myexit** (to exit the program). The declaration should also include **Q_OBJECT**. **Q_OBJECT** is a special object used by KDE to connect to the Qt Toolkit objects. The declaration for **Hellowin** will be placed in the **hellowin.h** header file, and all the member function definitions will be placed in the **hellowin.cpp** file. Figure 13 shows both the main window with its Exit and Hello buttons and the hello window displayed by this program.

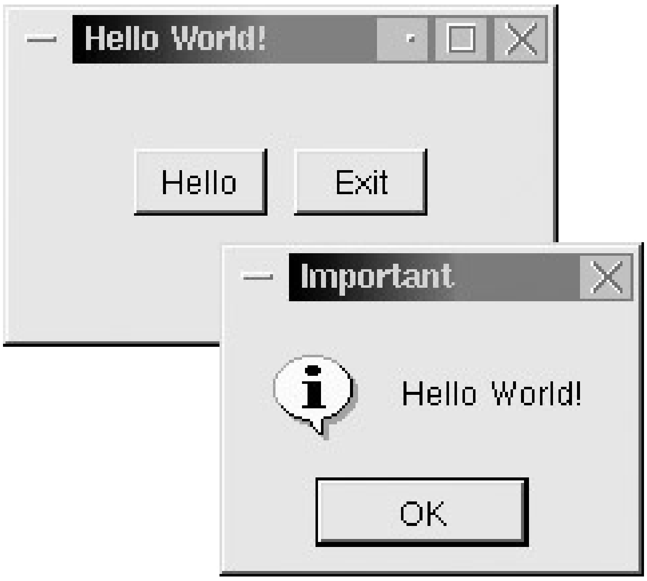


Figure 13: KDE Hellowin program

To compile this program, you first need to preprocess the **hellowin.h** header file with MOC. Then, you can compile the **hellowin.cpp** file. Notice that this file should include the **hellowin.moc** file, not the **hellowin.h** file. The compile operations are shown here:

```
moc –o hellowin.moc hellowin.h
g++ -lkdecore -lkdeui -lqt -o hellowin hellowin.cpp
```

Depending on your system, you may also need to specify the location of the KDE libraries and header files. These are usually held in a KDE directory specified in the KDEDIR system variable. Use the **-L** option with the **/lib** directory to specify the libraries, and **-I** options with the **/include** directory for header files, as in **-L$KDEDIR/lib** and **-I$KDEDIR/include**. If KDEDIR is not set, then check with your system administrator or with your Linux distribution manual for the location of the KDE libraries and header files. An example of the compile operation specifying the location of the KDE include and lib directories is shown here:

```
g++ -L$KDEDIR/lib -I$KDEDIR/include -lkdecore –lkdeui \
      -lqt -o hellowin hellowin.cpp
```

The displaying of the "Hello World" message will be handled by a **KMsgBox** object. This class implements a simple dialog box with messages and several buttons, if you want them. In addition to a simple message dialog box, **KMsgBox** also supports dialog boxes with Yes and No buttons and Yes, No, and Cancel buttons. Be sure to include **kmsgbox.h**.

The basic tasks for creating a **Hellowin** program are listed here.

- Create the **hellowin.h** header file that will hold the class definition of the **Hellowin** class.
- Include header files that contain class definitions for objects such as windows, buttons, and message boxes. In this case, include the header file for the main window, **ktmainwindow.h**, for a button, **qpushbutton.h**, and for a message box, **kmsgbox.h.**

- Define the **Hellowin** class, derived from the **KTMainWindow** class. Give the **Hellowin** class two slots (functions) called **myhello** and **myexit**. Define two pointers to buttons: **buttonhi** and **buttonExit**.
- The **myhello.cpp** file will hold the main program. Begin it by including header files and the **hellowin.moc** file, which needs to be separately generated by the MOC preprocessor.
- Define a constructor function for the **Hellowin** class, **Hellowin::Hellowin**. When an object of that class is defined, this function is automatically executed.
- Create a button object using **QPushButton**, and assign it to the **buttonhi** pointer. Set its size and then display it. It is connected to the **myhello** slot so that when it is clicked, the **myhello** function is executed.
- Create another button object using **QPushButton**, and assign it to the **buttonExit** pointer. Set its size and then display it. Connect it to the **myexit** slot so that when it is clicked, the **myexit** function will be executed.
- Define a **closeEvent** function for the **Hellowin** class. This function should simply end the program by invoking the **kapp->quit** function.
- Define the **myhello** function for the **Hellowin** class. This should display a separate message box with the message "Hello World".
- Define the **myexit** function for the **Hellowin** class, and set it to close the **Hellowin** window.
- In the main function, define a **Kapplication** called **myapp**. Define a **Hellowin** object called **mywin**. Then set the size of the object.
- Use the **setMainWidget** function for **myapp** to make the **mywin** object the main application window.
- Use **mywin.show** to show that window.
- Use **myapp.exec** to run the application.

The code for the **hellowin.h** file is shown next.

hellowin.h

```
#include <kapp.h>
#include <ktmainwindow.h>
#include <qpushbutton.h>
#include <kmsgbox.h>

class Hellowin : public KTMainWindow
        {
        Q_OBJECT
        public:
                Hellowin ();
                void closeEvent(QCloseEvent *);
        public slots:
                void myhello();
                void myexit();
        private:
                QPushButton *buttonhi;
                QPushButton *buttonExit;
        };
```

The following is the code for the **hellowin.cpp** file.

hellowin.cpp

```
#include <kapp.h>
#include <ktmainwindow.h>
#include <qpushbutton.h>
#include "hellowin.moc"
#include <kmsgbox.h>

Hellowin::Hellowin () : KTMainWindow()
        {
        buttonhi = new QPushButton("Hello", this);
        buttonhi->setGeometry(45,30,50,25);
        buttonhi->show();
        connect(buttonhi, SIGNAL(clicked()), this, SLOT(myhello()));

        buttonExit = new QPushButton("Exit", this);
        buttonExit->setGeometry(105,30,50,25);
        buttonExit->show();
        connect(buttonExit, SIGNAL(clicked()), this, SLOT(myexit()));
        }

void Hellowin::closeEvent(QCloseEvent *)
        {
        kapp->quit();
        }

void Hellowin::myhello()
        {
        KMsgBox::message(0,"Important","Hello World!");
        }

void Hellowin::myexit()
        {
        close();
        }

int main( int argc, char **argv )
        {
        KApplication myapp ( argc, argv, "Hello World!" );
        Hellowin mywin;
        mywin.setGeometry(100,100,200,100);

        myapp.setMainWidget( &mywin );
        mywin.show();
        return myapp.exec();
        }
```

Menus

To create menus, you create a KDE menu bar and then add Qt menus with their menu items to it. You will need to include the **qpopupmenu.h** header file for the menus and the **kmenubar.h** header file for the menu bar, as shown here:

```
#include <qpopupmenu.h>
#include <kmenubar.h>
```

You then define a menu bar object, or a pointer to one, and do the same for your menus. The class for a menu bar is **KMenuBar**, and the class for a menu is **QPopupMenu**. The following example defines pointers to a menu bar and a menu:

```
KMenuBar *mymenubar;
QPopupMenu *myfilemenu;
```

If you defined pointers, you can create the menu and menu bar objects with the **new** operation as shown here. **KMenuBar** takes as its argument its parent. When defined in a class like a window, where you want the class itself to be the parent, you use the **this** pointer.

```
mymenubar = new KMenuBar(this);
myfilemenu = new QPopupMenu;
```

You can then add the menu to the menu bar with the menu bar's **insertItem** member function. The first argument is the label you want displayed on the menu bar for the menu, and the second argument is the address of the menu object. The following example adds **myfilemenu** to **mymenubar**:

```
mymenubar->insertItem("File", myfilemenu);
```

Then, to add items to a particular menu, you use the menu object's **insertItem** member function. Its first argument is the label you want displayed for the item, and the next two arguments are references to a slot function to be executed when the item emits a signal. (This is the same as the slot arguments in the **connect** function.) The second argument for **insertItem** is the address of the object that holds the slot function, and the third argument is the slot function in that object to be executed.

The following example creates an Exit item in the **myfilemenu** menu and connects it to the **myexit** slot function in the current object (denoted by the **this** pointer):

```
myfilemenu->insertItem("Exit", this, SLOT(myexit()));
```

Qt Programming

KDE currently relies directly on the Qt Toolkit. Using just Qt objects, you can create an interface with a look and feel similar to KDE. You can create a Qt application using just Qt objects and the Qt libraries. This section provides a basic introduction to Qt programming. Both the KDE development site at **developer.kde.org** and the Qt Web site at **www.trolltech.com** provide very detailed documentation and tutorials on Qt programming. It is strongly recommended that you consult these resources for a detailed presentation of Qt programming and API references.

Web Chapter 44: Perl, Tcl/Tk, Expect, and Gawk

Perl, Tcl/Tk, Expect, and Gawk are scripting languages commonly used for customized applications on Linux. Your Red Hat Linux system installs these languages as part of its development package. Though beyond the scope of this book, a brief introduction to these languages is provided in this chapter.

Perl

The Practical Extraction and Report Language (Perl) is a scripting language originally designed to operate on files, generating reports and handling very large files. Perl was designed as a core program to which features could be easily added. Over the years, Perl's capabilities have been greatly enhanced. It can now control network connections, process interaction, and even support a variety of database management files. At the same time, Perl remains completely portable. A Perl script will run on any Linux system, as well as most other operating systems such as Unix, Windows, and Mac. Perl is also used extensively for implementing CGI scripts on Web sites. There are extensive and detailed man pages on Perl, discussing all aspects of the language with a great many examples. The man pages begin with the term **perl;** for example, **perlfunc** discusses the built-in Perl functions and **perlsyn** describes the different control structures. You can also use the **www.perl.com** site to access and search documentation including the reference manual, the online man pages, and FAQs.

There are extensive Internet resources for Perl. On the Perl Web site at **www.perl.com**, you can access documentation, software, newsgroups, and support. The site has programming and reference sections where you can access detailed FAQs on topics such as CGI, modules, and security. You can also access software archives and more detailed documentation.

Specialized Perl Web sites focus on programming, conferences, and reference resources. The Comprehensive Perl Archive Network (CPAN) maintains FTP sites that hold an extensive collection of utilities, modules, documentation, and current Perl distributions. You can also link to a CPAN site through the Perl Web sites. Several of the Perl Web sites are listed here:

www.perl.com
www.perlmongers.org
www.perl.org
republic.perl.com
www.perlreference.net
www.perl.com/CPAN/CPAN.html

The Perl advocacy group known as the perlmongers can be located at **www.perl.org** or **www.perlmongers.org**. The **republic.perl.com** site lets you join the Programming Republic of Perl. There are also several Usenet newsgroups that discuss different Perl issues. You can use them to post questions and check out current issues. Here is a listing of the current newsgroups:

comp.lang.perl.announce
comp.lang.perl.misc
comp.lang.perl.modules
comp.lang.perl.tk

Perl Scripts

Usually, Perl commands are placed in a file that is then read and executed by the **perl** command. In effect, you are creating a shell in which your Perl commands are executed. Files containing Perl commands must have the extension **.pl**. This identifies a file as a Perl script that can be read by the **perl** command. There are two ways that you can use the **perl** command to read Perl scripts. You can enter the **perl** command on the shell command line, followed by the name of the Perl script. Perl will read and execute the commands. The following example executes a Perl script called **hello.pl**:

```
$ perl hello.pl
```

You can also include the invocation of the **perl** command within the Perl script file, much as you would for a shell script. This automatically invokes the Perl shell and will execute the following Perl commands in the script. The path **/usr/bin/perl** is the location of the **perl** command on the OpenLinux system. On other systems, it could be located in **/usr/local/bin** directory. The command **which perl** will return the location of Perl on your system. Place the following shell instruction on the first line of your file:

```
#!/usr/bin/perl
```

Then, to make the script executable, you would have to set its permissions to be executable. The **chmod** command with the **755** option sets executable permissions for a file, turning it into a program that can be run on the command line. You only have to do this once per script. You do not have to do this if you use the **perl** command on the command line, as noted previously. The following example sets the executable permissions for the **hello.pl** script:

```
$ chmod 755 hello.pl
```

As in C, Perl commands end with a semicolon. There is a **print** command for outputting text. Perl also uses the same escape sequence character to output newlines, **\n**, and tabs, **\t**. Comments are lines that begin with a **#**. The following is an example of a Perl script. It prints out the word "hello" and a newline. Notice the invocation of the **perl** command on the first line:

```
helloprg

#!/usr/bin/perl

print "hello \n";

$ helloprg
hello
```

Perl Input and Output: <> and print

A Perl script can accept input from many different sources. It can read input from different files, from the standard input, and even from pipes. Because of this, you have to identify the source of your input within the program. This means that, unlike with Gawk but like with a shell program, you have to explicitly instruct a Perl script to read input. A particular source of input is identified by a file handle, a name used by programs to reference an input source such as a particular file. Perl already sets up file handles for the standard input and the standard output, as well as the standard error. The file handle for the standard input is **STDIN**.

The same situation applies to output. Perl can output to many different destinations, whether they be files, pipes, or the standard output. File handles are used to identify files and pipes when used for either input or output. The file handle **STDOUT** identifies the standard output, and **STDERR** is the file handle for the standard error. We shall first examine how Perl uses the standard input and output, and later discuss how particular files are operated on.

Perl can read input from the standard input or from any specified file. The command for reading input consists of the less-than (<) and greater-than (>) symbols. To read from a file, a file handle name is placed between them, **<MYFILE>**. To read from the standard input, you can simply use the **STDIN** file handle, **<STDIN>**, which is similar to the **read** command in the BASH shell programming language.

```
<STDIN>
```

To use the input that **<STDIN>** command reads, you assign it to a variable. You can use a variable you define or a default variable called **$_,** as shown in the next example. **$_** is the default for many commands. If the **print** command has no argument, it will print the value of **$_**. If the **chomp** command has no argument, it operates on **$_**, cutting off the newline. The **myread** script that follows illustrates the use of **$_** with the standard input:

```
myread

#!/usr/bin/perl
# Program to read input from the keyboard and then display it.

$_ = <STDIN>; #Read data from the standard input
print "This is what I entered: $_"; #Output read data as part of a string

$ myread
larisa and aleina
This is what I entered: larisa and aleina
```

You can use the **print** command to write data to any file or to the standard output. File handle names are placed after the **print** command and before any data such as strings or variables. If no file handle is specified, **print** outputs to the standard output. The following examples both write the "hello" string to the standard output. The explicit file handle for the standard output is **STDOUT**. If you do not specify an argument, **print** will output whatever was read from the standard input.

```
print STDOUT "hello";
print "hello";
```

Tip A null file handle, **<>**, is a special input operation that will read input from a file listed on the command line when the Perl script is invoked. Perl will automatically set up a file handle for it and read. If you list several files on the command line, Perl will read the contents of all of them using the null file handle. You can think of this as a **cat** operation in which the contents of the listed files are concatenated and then read into the Perl script.

Perl File Handles

You use the **open** command to create a file handle for a file or pipe. The **open** command takes two arguments: the name of the file handle and the filename string. The name of the file handle is a name you make up. By convention, it is uppercase. The filename string can be the name of the file or a variable that holds the name of the file. This string can also include different modes for opening a file. By default, a file is opened for reading. But you can also open a file for writing, or for appending, or for both reading and writing. The syntax for **open** follows:

```
open ( file-handle, filename-string);
```

In the next example, the user opens the file reports, calling the file handle for it **REPS**:

```
open (REPS, "reports");
```

Often the filename will be held in a variable. You then use the **$** with the variable name to reference the filename. In this example, the filename "reports" is held in the variable **filen**:

```
filen = "reports";
open (REPS, $filen );
```

To open a file in a specific mode such as writing or appending, you include the appropriate mode symbols in the filename string before the filename, separated by a space. The different mode symbols are listed in Table 1. The symbol > opens a file for writing, and +> opens a file for both reading and writing. In the next example, the reports file is opened for both reading and writing:

```
open (REPS, "+> reports");
```

Table 1: Perl: File Operations and Command Line Options

Perl Command Line Options	**Description**
-e	Enter one line of a Perl program.
-n	Read from files listed on the command line.
-p	Output to standard output any data read.
Perl File Commands	
open(*file-handle***,** *permission-with-filename***)**	Open a file.
close(*file-handle***)**	Close a file.
filename	Read from a file.
STDIN	Read from the standard input.
	Read from files whose filenames are provided in the argument list when the program was invoked.
print file-*handle text;*	Write to a file. If no file handle is specified, write to standard output. If no text is specified, write contents of **$_**.
printf handle " *Format-str* **",** *values ;*	Write formatted string to a file. Use conversion specifiers to format values. If no file handle is specified, write to standard output. If no values are specified, use contents of **$_**.
sprintf *str-var* **"** *Format-str* **",** *values ;*	Write formatted values to a string. Use conversion specifiers to format values. If no values are specified, use contents of **$_**.
Permissions for Opening Files	

Table 1: Perl: File Operations and Command Line Options

Perl Command Line Options	**Description**
filename	Read-only.
filename	Write-only.
+ *filename*	Read and write.
filename	Append (written data is added to the end of the file).
command |	An input pipe, reading data from a pipe.
| *command*	An output pipe, sending data out through this pipe.

If you are using a variable to hold the filename, you can include the evaluated variable within the filename string, as shown here:

```
open (REPS, "+> $filen");
```

To read from a file using that file's file handle, you simply place the file handle within the < and > symbols. **<REPS>** reads a line of input from the reports file. In the **myreport** program, the reports file is opened and its contents are displayed.

myreport.pl

```
#!/usr/bin/perl
# Program to read lines from the reports file and display them

open(REPS, "< reports"); # Open reports file for reading only
while ( $ldat = <REPS> ) # Read a line from the reports file
 {
 print $ldat; # Display recently read line
 }
close REPS; # Close file
```

Perl also has a full set of built-in commands for handling directories. They operate much like the file functions. The **opendir** command opens a directory, much as a file is opened. A directory handle is assigned to the directory. The **readdir** command will read the first item in the directory, though, when in a list context, it will return all the file and directory names in that directory. **closedir** closes the directory, **chdir** changes directories, **mkdir** creates directories, and **rmdir** removes directories.

Perl Variables and Expressions

Perl variables can be numeric or string variables. Their type is determined by context—the way they are used. You do not have to declare them. A variable that is assigned a numeric value and is used in arithmetic operations is a numeric variable. All others are treated as strings. To reference a variable in your program, you precede it with a **$** symbol, just as you would for a shell variable.

You can use the same set of operators with Perl variables as with C variables—with the exception of strings. Strings use the same special comparison terms as used in the Bourne shell, not the standard comparison operators. Those are reserved for numeric variables. However, other operators such as assignment operators work on both string and numeric variables. In the next example, the variable **myname** is assigned the string "Larisa". The assignment operator is the = symbol (see Table 2).

```
$myname = "Larisa";
```

For a numeric variable, you can assign a number. This can be either an integer or a floating-point value. Perl treats all floating-point values as double precision.

```
$mynum = 45;
$price = 54.72;
```

Perl also supports arithmetic expressions. All the standard arithmetic operators found in other programming languages are used in Perl. Expressions can be nested using parentheses (see Table 2). Operands can be numeric constants, numeric variables, or other numeric expressions. In the following examples, **$mynum** is assigned the result of an addition expression. Its value is then used in a complex arithmetic expression whose result is assigned to **$price**.

```
$mynum = 3 + 6;
$price = ( 5 * ($num / 3);
```

Table 2: Arithmetic: , Relational (Numeric), and Assignment Operators

Arithmetic Operators	**Function**
*	Multiplication
/	Division
+	Addition
-	Subtraction
%	Modulo—results in the remainder of a division
**	Power
Relational Operators	
	Greater than
	Less than
=	Greater than or equal to
=	Less than or equal to
==	Equal in let
!=	Not equal
Increment Operators	
++	Increment variable by one
«—	Decrement variable by one
Arithmetic Assignment Operators	
+=	Increment by specified value

Table 2: Arithmetic: , Relational (Numeric), and Assignment Operators

Arithmetic Operators	**Function**
-=	Decrement by specified value
/=	Variable is equal to itself divided by value
*=	Variable is equal to itself multiplied by value
%=	Variable is equal to itself remaindered by value

Perl supports the full range of assignment operators found in Gawk and C. The ++ and «— operators will increment or decrement a variable. The += and **-=** operators and their variations will perform the equivalent of updating a variable. For example, **i++** is the same as **i = i + 1**, and **i += 5** is the same as **i= i + 5**. Increment operations such as **i++** are used extensively with loops.

You can easily include the value of a variable within a string by simply placing the variable within it. In the following example, the value of **$nameinfo** would be the string "My name is Larisa \n":

```
print "The number of items is $mynum \n"
$nameinfo = "My name is $myname \n"
```

To assign data read from a file to a variable, just assign the result of the read operation to the variable. In the next example, data read from the standard input is assigned to the variable **$mydata**:

```
$mydata = <STDIN>;
```

When reading data from the standard input into a variable, the carriage return character will be included with the input string. You may not want to have this carriage return remain a part of the value of the variable. To remove it, you can use the **chomp** command, which removes the last character of any string. With data input from the keyboard, this happens to be the carriage return.

```
chomp $myinput;
```

In the next example, the user inputs his or her name. It is assigned to the **myname** variable. The contents of **myname** is then output as part of a string. **chomp** is used to remove the carriage return from the end of the **$myname** string before it is used as part of another string.

```
readname.pl

#!/usr/bin/perl
$myname = <STDIN>;
chomp $myname;

print "$myname just ran this program\n";

$ myread.pl
larisa Petersen
larisa Petersen just ran this program
```

Arrays and Lists

In Perl, you create an array by assigning it a list of values. A list in Perl consists of a set of values encased in parentheses and separated by colons. The following example is a list of four values:

```
( 23, 41, 92, 7)
```

You assign this list to the array you wish to create, preceding the array name with an @ sign. This assignment will initialize the array sequentially, beginning with the first value in the list:

```
@mynums = (23, 41, 92, 7);
```

Once the array has been created, you can reference its individual elements. The elements start from 0, not 1. The **mynums** array has four elements, numbered from 0 to 3. You can reference individual elements using an index number encased within brackets. **[0]** references the first element, and **[2]** references the third element. The following example prints out the first element and then the fourth element. Notice that the array name is prefixed with a **$**.

```
print $mynums[0] ;
print $mynums[2] ;
```

You can change the value of any element in the array by assigning it a new value. Notice that you use a **$**, not an @ sign, preceding an individual array element. The @ sign references the entire array and is used when you are assigning whole lists of values to it. The **$** sign references a particular element, which is essentially a variable.

```
$mynums[2] = 40;
```

There is no limit to the number of elements in the array. You can add more by simply referencing a new element and assigning it a value. The following assignment will add a fifth element to the **mynums** array:

```
$mynums[4] = 63;
```

Each array will have a special variable that consists of a # and the name of the array. This variable is the number of elements currently in the array. For example, **#mynums** holds the number of elements in the **mynums** array. The following example prints out the number of elements. Notice the preceding **$**.

```
print "$#mynums";
```

When assigning a list to an array, the values in a list do not have to be of the same type. You can have numbers, strings, and even variables in a list. Similarly, elements of the array do not have to be of the same type. One element could be numeric, and another a string. In the next example, the list with varied elements is assigned to the **myvar** array:

```
@myvar = ( "aleina", 11, 4.5, "a new car");
```

You can reference the entire set of elements in an array as just one list of values. To do this, you use the array name prefixed by the @ sign. The following example will output all the values in the **mynums** array:

```
print @mynums;
```

The @ is used here instead of the **$**, because the array name is not a simple variable. It is considered a list of values. Only the individual elements are variables. This means that to just reference all the values in an array, you use the @ sign, not the **$**. This is even true when you want to assign one array to another. In the next example, the values of each element in **mynums** are assigned to corresponding elements in **newnums**. Notice the @ used for **mynums**. You can think of **@mynums** as evaluating to a list of the values in the **mynums** array, and this list being then assigned to **newnums**.

```
@newnums = @mynums;
```

Perl Control Structures

Perl has a set of control structures similar to those used in the Gawk, TCSH shell, and C programming languages. Perl has loops with which you can repeat commands, and conditions that allow you to choose among specified commands. For the test expressions, there are two different sets of operators for use with strings and numeric values. Table 2 lists the numeric relational operators, and Table 3 lists the string operators. You can also use pattern operations that allow the use of regular expressions. Table 4 lists the Perl control structures with their syntax.

Table 3: String: , Logical, File, and Assignment Operators

String Comparisons	**Function**
gt	Greater than.
lt	Less than.
ge	Greater than or equal to.
le	Less than or equal to.
eq	Equal.
ne	Not equal.
Logical Operations	
expression **&&** *expression* *expression* **and** *expression*	The logical AND condition returns a true 0 value if both expressions return a true 0 value; if one returns a nonzero value, the AND condition is false and also returns a nonzero value. Execution stops at the first false expression. The **and** operation is the same as **&&** but has a lower precedence.
expression || *expression* *expression* **or** *expression*	The logical OR condition returns a true 0 value if one or the other expression returns a true 0 value; if both expressions return a nonzero value, the OR condition is false and also returns a nonzero value. Evaluation stops at the first true expression. The **or** operation is the same as || but has a lower precedence.

Table 3: String: , Logical, File, and Assignment Operators

String Comparisons	**Function**
! *command* **not** *command*	The logical NOT condition inverts the true or false value of the expression. The **not** operation is the same as **!** but has a lower precedence.
File Tests	
-e	File exists.
-f	File exists and is a regular file.
-s	File is not empty.
-z	File is empty, zero size.
-r	File is readable.
-w	File can be written to, modified.
-x	File is executable.
-d	Filename is a directory name.
-B	File is a binary file.
-T	File is a text file.
Assignment Operator	
=	Assign a value to a variable.

Table 4: Perl: Conditions, Loops, and Functions

Control Stucture	**Description**

LABEL:{
statements;
}
	Block is a collection of statements enclosed within opening and closing braces. The statements are executed sequentially. The block can be labeled.

Conditional Control Structures: if, else, elsif, case	

if(*expression* **)** {
statements;
}
	if executes statements if its test expression is true. Statements must be included within a block.

if(*expression* **)** {
statements;
}
else(*expression* **)** {
statements;
}
	if-else executes statements if the test expression is true; if false, the **else** statements are executed.

if(*expression* **)** {
statements;
}
elsif(*expression* **)** {
statements;
}
else(*expression* **)** {
statements;
}
	elsif allows you to nest **if** structures, enabling selection among several alternatives; at the first true **if** expression, its statements are executed and control leaves the entire **elsif** structure.

Table 4: Perl: Conditions, Loops, and Functions

Control Stucture	**Description**

unless(*expression* **)** {
statements;
}
	unless executes statements if its test expression is false.

test **?** *stat1* **:** *stat2*	Conditional expression. If true, executes *stat1,* **else** *stat2.*

*LABEL***:{**
if(*expr***){***statements;***last** *LABEL***}***;*
}
	Simulate a **switch** structure by using listed **if** statements within a block with the **last** statement referencing a label for the block.

Loop Control Structures: while, until, for, foreach	

*LABEL:***while(** *expression* **)** {
statements;
}
	while executes statements as long as its test expression is true. LABEL is optional.

do{
statements;
} until(*expression* **)**
	until executes statements as long as its test expression is false.

foreach *variable* **(** *list-values***)**
{
statements;
}
	foreach is designed for use with lists of values such as those generated by an array; the variable operand is consecutively assigned the values in the list.

for(*init-expr; test-expr; incr-expr***)**
{
statements;
}
	The **for** control structure executes statements as long as *test-expr* is true. The first expression, *init-expr*, is executed before the loop begins. The third expression, *incr-expr,* is executed within the loop after the statements.

*LABEL***:** *block-or-loop*	Label a block or loop. Used with the **next**, **last**, and **redo** commands.
Functions	
sub *function-name* **;**	Declare a function.

sub *function-name* **{**
statements;
}
	Define a function with the name *function-name.*

& *function-name***(***arg-list***)**	Call a function with arguments specified in the argument list.
@@_	Holds the values of arguments passed to the current function. **$_** and index references an argument. **$_**[0] is the first argument.
$#_	Number of arguments passed to the current function.

Test Expressions

Perl has different sets of operators for numeric and string comparisons. You have to be careful not to mix up the different operators. The string operators are two-letter codes similar to those used in the BASH shell. For example, the **eq** operator tests for the equality of two strings, and the **gt** operator tests to see if one is greater than the other. Numeric comparisons, on the other hand, use as operators symbols similar to those found in programming languages. For example, > stands for greater than, and == tests for equality. These are essentially the same comparison operators used for the C programming language.

There are two important exceptions: patterns and string patterns. The string-pattern operator, =~, tests for a pattern in a string variable. The right-hand operand is the pattern, and the left-hand operand is the string. The pattern can be any regular expression, making this a very flexible and powerful operation.

Patterns perform pattern matching on either a string or the contents of the **_$** special variable. The pattern operator consists of two slashes that enclose the pattern searched for, ***/pattern/***. The pattern can be any regular expression. Regular expression are discussed in detail in the section on pattern matching.

Perl supports the AND, OR, and NOT logical operations. There are two implementations of these operations: the standard operators and those with list processing capabilities. The standard logical operators are **&&**, ||, and **!**. The **&&** operator implements an AND operation, || an OR, and **!** a NOT. They take as their operands expressions, just as they do in the C programming language. Their syntax is as follows:

```
(expression) && (expression)
(expression) || (expression)
!(expression)
```

In the case of the logical AND, **&&**, if both commands are successful, the logical command is true. For the logical OR, ||, if either command is successful, the OR is true. You can extend these commands with added **&&** or || commands, creating complex AND or OR operations. The logical commands allow you to use logical operations as your test commands in control structures. You can also use them independently.

The logical operators are usually used in test expressions for control structures such as **while** and **if**. But they can also be used independently as their own statements. In effect, they provide a simple way to write a conditional operation. In Perl scripts, you may often see an OR operation used with a file **open** command. In an OR operation, if the first expression fails, the second one is checked. If that second one is the **die** command to end the program, in effect there is an operation to end the program if the file **open** operation fails (the first expression fails).

```
open (REPS, "+> $filen") or die "Can't open $filen";
```

The AND operation works similarly, except that if the first expression is true, the second one is checked. The following looks for an empty line and, if it finds it, prints the message:

```
/^$/ && print "Found empty line";
```

Loops

Perl loops are the **while**, **do-until**, **for**, and **foreach** loops. The **while** loop is the more general-purpose loop, whereas the **for** and **foreach** loops provide special capabilities. The **foreach** is particularly helpful in processing lists and arrays. The **while**, **do-until**, and **for** loops operate much like their counterparts in the C programming language. The **for** loop, in particular, has the same three expression formats as the C **for** loop. The **foreach** loop is similar to its counterpart in the C shell, able to easily handle lists of items.

You can easily adapt the **while** loop for use with arrays. The variable used to control a loop can also be used, inside the loop, to index an array. In the next example, the elements of the **title** array are assigned the value of each title. Then, the contents of each element are printed out using a **for** loop. Notice that the **$#num** holds the count of elements in the array. **$#num** is used as the upper bound in the **for** loop test expression in order to check when the loop should stop.

```
titlearr.pl

#!/usr/bin/perl
# Program to define and print out a scalar array

@title = ( "Tempest", "Iliad", "Raven"); # define array with 3 elements

for($i = 0; $i <= $#title; $i++) # Loop through array, $#title is size
 {
 print "$title[$i] \n"; # Print an element of the title array
 }
```

```
$ titlearr.pl
Tempest
Iliad
Raven
```

The **foreach** loop is useful for managing arrays. In the **foreach** loop, you can use the array name to generate a list of all the element values in the array. This generated list of values then becomes the list referenced by the **foreach** loop. You can also specify a range of array elements, using only those values for the list, or a set of individual elements. In the next example, the array name **@mylist** is used to generate a list of its values that the **foreach** loop can operate on, assigning each one to **$mynum** in turn:

```
mynumlist.pl

#!/usr/bin/perl
# Program to use foreach to print out an array

@mylist = ( 34, 21, 96, 85); # define array of 4 elements

foreach $mynum ( @mylist ) # Assign value of each element to $mynum
 {
 print "$mynum \n";
 }
```

Using the **@ARGV** array, you can specify the command line arguments as a list of values. The arguments specified on the command line when the program was invoked become a list of values referenced by the **foreach** loop. The variable used in the **foreach** loop is set automatically to each argument value in sequence. The first time through the loop, the variable is set to the value of the first argument. The second time, it is set to the value of the second argument, and so on.

Tip The number of arguments that a user actually enters on the command line can vary. The **#ARGV** variable will always hold the number of arguments that a user enters. This is the number of elements that are in the **ARGV** array. If you want to reference all the elements in the **ARGV** array using their indexes, you will need to know the number of elements, **#ARGV**. For example, to use the **foreach** loop to reference each element in the **ARGV** array, you would use the **..** operator to generate a list of indexes. **0.. $#ARGV** generates a list of numbers beginning with 0 through to the value of **$#ARGV**.

Conditions: if, elsif, unless, and switch

Perl supports if-else operations much as they are used in other programming languages. The **if** structure with its **else** and **elsif** components allows you to select alternative actions. You can use just the **if** command to choose one alternative, or combine that with **else** and **elsif** components to choose among several alternatives. The **if** structure has a test expression encased in parentheses followed by a block of statements. If the test is true, the statements in the block are performed. If not, the block is skipped. Unlike in other programming languages, only a block can follow the test, and any statements, even just one, must be encased with it. The following example tests to see if an **open** operation on a file was successful. If not, it will execute a **die** command to end the program. The NOT operator, **!**, will make the test true if **open** fails, thereby executing the **die** command.

```
if (!(open (REPS, "< $filen"))) {
 die "Can't open $filen";
}
else {
 print "Opened $filen successfully";
 }
```

Tcl, Tk, and Expect

Tcl is general-purpose command language developed by John Ousterhout in 1987 at the University of California, Berkeley. Originally designed to customize applications, it has become a fully functional language in its own right. As with Perl and Gawk, you can write Tcl scripts, developing your own Tcl programs. Tcl is a very simple language to use.

TK and Expect are Tcl applications that extend the capabilities of the language. The Tk application allows easy development of graphical interactive applications. You can create your own windows and dialog boxes with buttons and text boxes of your choosing. The Expect application provides easy communication with interactive programs such as FTP and telnet.

Tcl is often used in conjunction with Tk to create graphical applications. Tk is used to create the graphical elements such as windows, and Tcl performs the programming actions such as managing user input. Like Java, Tcl and Tk are cross-platform applications. A Tcl/Tk program will run on any platform that has the Tcl/Tk interpreter installed. Currently, there are Tcl/Tk versions for Windows, the Mac, and Unix systems, including Linux. You can write a Tcl application on Linux and run the same code on Windows or the Mac. The new versions of

Tcl and Tk 8.0 even support a local look and feel for GUI widgets using Mac-like windows on the Mac, but Windows-like windows under Windows 98.

Tcl is an interpreted language operating, like Perl, within its own shell; **tclsh** is the command for invoking the Tcl shell. Within this shell, you can then execute Tcl commands. You can also create files within which you can invoke the Tcl shell and list Tcl commands, effectively creating a Tcl program. A significant advantage to the Tcl language and its applications is the fact that it is fully compatible with the C programming language. Tcl libraries can be incorporated directly into C programs. In effect, this allows you to create very fast compiled versions of Tcl programs.

When you install Tk and Tcl on your system, Man pages for Tcl/Tk commands are also installed. Use the **man** command with the name of the Tcl or Tk command to bring up detailed information on that command. For example, **man switch** displays the manual page for the Tcl **switch** command, and **man button** displays information on the Tk button widget. Once you have installed Tk, you can run a demo program called **widget** that shows you all the Tk widgets available. The **widget** program uses Tcl/Tk sample programs and can display the source code for each. You can find the **widget** program by changing to the Tk **demos** directory as shown here. (**tk*** here matches on directory names consisting of **tk** and its version number, **tk4.1** for version 4.1, and **tk8.0** for version 8.0.)

```
cd /usr/lib/tk*/demos
```

From the Xterm window, just enter the command **widget**. You can also examine the individual demo files and modify them as you wish. If you have installed a version of Tk yourself into the **/usr/local/bin** directory rather than **/usr/bin**, the **demos** directory will be located in **/usr/local/lib/tk***.

Tcl/Tk Extensions and Applications

Currently, both Tcl and Tk are being developed and supported as open-source projects by Tcl Core Team. The current release of both Tcl and Tk is 8.0, though version 8.1 will be ready soon. Current versions of Tcl and Tk are available free of charge from the Tcl Developer Xchange Web site, http://dev.scriptics.com. Also available on this site is extensive documentation for each product in PostScript, Adobe PDF, and HTML formats. The HTML documentation can be viewed online. RPM packaged versions can also be found at distribution sites such as ftp.redhat.com. You will need both Tcl and Tk RPM packages as well as the development packages for each.

Tcl/Tk has been enhanced by extensions that increase the capabilities of the language. Several commonly used extensions are TclX, [incr Tcl], and Oratcl. All these are currently available though links on the Tcl Developer Xchange Web site. Access the Tcl Software Resource page and from there, you can access the Tcl Extensions page, listing those currently available (**dev.scriptics.com/software**). Most you can download and install for free. Most are also available at http://sourceforge.net. TclX extends capabilities such as file access and time and date manipulation, many of which have been incorporated into recent Tcl releases. [incr Tcl] supports the easy implementation of higher-level widgets, using an object-oriented programming structure. BLT adds graph and bar widgets to Tk. Sybtcl and Oratcl implement database interfaces to the Sybase and Oracle databases. TclDP provides support for distributed programming across a network. With the Scotty Network Management Tools, you can develop network management applications. The TrfCrypt extension adds encryption that was removed from the standard Tcl/Tk release to make it exportable.

Numerous Tcl/Tk applications, development tools, and utilities are freely available for download from Internet sites. You can link to most of these through the Tcl Developer Xchange site or through **www.tcltk.com** site. Most products are also open-source projects available at **http://sourceforge.net**. If a site does not provide an RPM-packaged version of its software, be sure to check the appropriate distribution site, such as **ftp.redhat.com**. Of particular note is the Tcl/Tk Web page plug-in that allows you to embed Tcl/Tk programs within a Web page. Such embedded Tcl/Tk programs are called Tclets. There is also a Tcl/Tk Web server called TclHttpd that can be easily embedded in applications, making them Web capable. You can configure and modify the server using Tcl/Tk commands. Several GUI builders are also available that let you build graphical user interfaces (GUIs). Free Visual Tcl, SpecTcl, VisualGIPSY, and XF SpecTcl are GUI builders that have a window-based interface with menus and icons for easily creating Tcl/Tk widgets. They can generate both Tcl/Tk and Java code, and can be used to create standalone Tcl/Tk applications or Web Tclets. There are also editors (TextEdit), desktops (TK Desk), multimedia (MBone), and specialized applications like Tik, a Tcl/Tk implementation of AOL Instant Messenger.

Tcl

Tcl is a simple-to-use programming language. Its statements consist of a command followed by arguments, though it also has a complete set of control structures including **while** and **for** loops. Commands can be terminated either by a semicolon or by a newline. You can think of a Tcl command as a function call where the command name operates like a function name, followed by arguments to the function. However, unlike with a function call, there are no parentheses or commas encasing the arguments. You simply enter the command name and then its arguments, separated only by spaces. A newline entered after the last argument will end the statement.

You can see the features in this format very clearly in the Tcl assignment command, **set**. To assign a value to a variable, you first enter the assignment command **set**. Then enter the name of the variable, followed by the value to be assigned. The command name, variable, and value are separated only by spaces. The newline at the end of the line ends the statement. The following statement assigns a string **"larisa"** to the variable **myname**, and the next statement assigns the integer value 11 to the variable **age**:

```
set myname "larisa"
set age 11
```

Note Variable types are determined by their use. A variable assigned an integer will be considered an integer, and one assigned a string will be a character array.

The Tcl Shell and Scripts: tclsh

You execute Tcl commands within the Tcl shell. You can do this interactively, entering commands at a Tcl shell prompt and executing them one by one, or you can place the commands in a script file and execute them all at once. To start up the Tcl shell, you enter the command **tclsh**. This starts up the Tcl shell with the % prompt. You can then enter single Tcl commands and have them evaluated when you press ENTER. You leave the Tcl shell by entering either an **exit** command or a CTRL-D.

```
$ tclsh
% set age 11
% puts $age
11
% exit
$
```

You can run a Tcl script either as a standalone program or as a file explicitly read by the Tcl shell command **tclsh**. A Tcl script has the extension **.tcl**. For example, the **myread.tcl** Tcl script would be read and executed by the following command:

```
$ tclsh myread.tcl
```

To create a standalone script that operates more like a command, you need to invoke the **tclsh** command within the script. You can do this using an explicit pathname for the **tclsh** command. This is placed on the first line of the script.

```
#!/usr/bin/tclsh
```

Expressions

Expressions are also handled as commands. The command **expr** evaluates an expression and returns its resulting value as a string. It takes as its arguments the operands and operator of an expression. Tcl supports all the standard arithmetic, comparison, and logical operators. The result of an arithmetic expression will be the same form as its operands. If the operands are real numbers, the result will be a real number. You can mix operands of different types, and Tcl will convert one to be the same as the other. In the case of real and integer operands, the integer will be converted to a real. In the next statement, the addition of 4 and 3 is evaluated by the **expr** command. The following statement multiplies 25 and 2:

```
expr 4 + 3
expr 25 * 2
```

Tip The resulting value returned by any Tcl command is always a string. In the case of arithmetic operations, the arithmetic value is converted first to a string, which is then returned by the **expr** command.

Embedded Commands

You can combine commands by embedding one within the other. Embedding is commonly used for assigning the result of an expression to a variable. This involves two commands, the **set** command to perform the assignment and the **expr** command to evaluate an expression. You embed commands using brackets. An embedded command is another Tcl command whose result is used as an argument in the outer Tcl command. The embedded command is executed first, and its result is used as the argument to the outer command. The following statement assigns the result of the arithmetic operation, 25 * 2, to the variable **num**. **expr 25 * 2** is a command embedded within the **set** command. First the embedded command is executed, and its result, "50", is assigned to the variable **num**.

```
set num [expr 25 * 2]
```

Variables

Tcl supports numeric and string variables as well as arrays, including associative arrays. All variables hold as their contents a string. However, though the content of a variable is a string, that string can be used as an integer or real value in an arithmetic expression, provided that the string consists of numbers. Whenever such a variable is used as an operand in an arithmetic expression, its contents are first converted to an integer or real value. The operation is performed on the arithmetic values, and the result returned by **expr** is then converted back to a string. This means that you do not have to worry about declaring the type of variable, or even defining a variable. All variables are automatically defined when they are first used in a statement.

As we have seen, variables can be assigned values using the **set** command. The **set** command takes as its argument the variable name and the value assigned. A variable's name can be any set of alphabetic or numeric characters and the underscore. Punctuation and other characters are not allowed.

When you need to use the value of a variable within a statement, you need to evaluate it. Evaluating a variable substitutes its name with its value. The **$** placed before a variable name performs such an evaluation. To use a variable's value as an operand in an expression, you need to evaluate the variable by preceding its name with the **$**. In the next example, the value 5 is assigned to the **mynum** variable. Then **mynum** is evaluated in an expression, **$mynum**, providing its value, 5, as an operand in that expression.

```
set mynum 5
expr 10 * $mynum
```

Should you want to make the value of a variable part of string, you only need to evaluate it within that string. The value of the variable becomes part of the string. In the following statement, the value of the variable **myname** is used as part of a string. In this case, the string will be **"My name is Larisa"**.

```
set myname "Larisa"
set mystr "My name is $myname"
```

Certain commands are designed to operate on variables. The **append** command concatenates a string to a variable. The **incr** command will increment an integer, and the **unset** command will undefine a variable. The different commands that operate on variables are listed in Table 5.

Table 5: Assignments: and Variables

Commands	**Description**
set	Assign a value to a variable
global	Declare global variables
incr	Increment a variable by an integer value
unset	Delete variables
upvar	Reference a variable in a different scope
variable	Declare namespace variables

Table 5: Assignments: and Variables	
Commands	**Description**
array	Array access operations like searches
expr	Math expressions

Arrays

Array elements are defined and assigned values using the **set** command with the index encased in parentheses. The following example assigns the string **"rain"** as the second element in the **weather** array:

```
set weather(2) rain
```

You can then reference the element using its index encased in parentheses with the array name preceded with a **$**.

```
puts $weather(2)
rain
```

Tcl allows the use of any word string as an index, in effect supporting associative arrays. The string used as the index is encased within parentheses next to the array name. The following statements add two elements to the **city** array with the index strings **Napa** and **Alameda**:

```
set city(Napa) 34
set city(Alameda) 17
```

Control Structures

Tcl has a set of control structures similar to those used in the Perl, Gawk, C shell, and C programming languages. Tcl has loops with which you can repeat commands and conditions that allow you to choose among specified commands. Table 6 lists the Tcl control structures. Control structures in Tcl often make use of a block of Tcl commands. A block of commands consists of Tcl commands enclosed in braces. The opening brace ({) will begin on the same line as that of the control structure that uses it. On following lines, there can be several Tcl commands, each on its own line. The block ends with a closing brace (}) on a line by itself. A block is literally an argument to a Tcl command. The block is passed to a control structure command, and the control structure will execute the commands in that block.

if

The **if** control structure allows you to select alternative actions. The **if** command takes two arguments, a test expression and a Tcl command or block of commands. Each is encased in its own set of braces. The test expression is used to determine if the Tcl commands will be executed. If the test expression is true, the commands are performed. If not, the commands are skipped. Below is the syntax for the **if** structure. You can specify alternative commands to execute if the expression is false by attaching an **else** block with those Tcl commands. You can nest **if** structures using the **elseif** command.

Table 6: Tcl: Control Structures

Control Structures	**Description**
if	Conditional command, extend with else and **elseif** blocks
switch	Switch selection structure
while	The **while** loop
for	The **for** loop, like the C **for** loop
foreach	Loop through a list, or lists, of values
break	Forced loop exit

```
if {test-expression} {
 Tcl commands
 } elseif {test-expression} {
 Tcl commands
 } else {
 Tcl commands
 }
```

Note Keep in mind that the **else** and **elseif** keywords must be on the same line as the closing brace of the previous block, as in **} elseif { test-expression*}* {** . The opening brace of the next block must also be on the same line.

switch

The **switch** structure chooses among several possible alternatives. The choice is made by comparing a string value with several possible patterns. Each pattern has its own block of Tcl commands. If a match is found, the associated block is executed. The **default** keyword indicates a pattern that matches anything. If all of the other matches fail, the block associated with the **default** keyword is executed. The **switch** structure begins with the keyword **switch**, the options prefixed with **-**, and the string pattern to be matched, followed by a block containing all the patterns with their blocks. The syntax for the **switch** structure is described next:

```
 switch -options string-pattern {
 pattern {
 Tcl commands
 }
 pattern {
 Tcl commands
 }
 default {
 Tcl commands
 }
 }
```

Options specify the pattern-matching capabilities. The following options are supported:

-exact	Use exact matching when comparing string to a pattern. This is the default.
-glob	When matching string to the patterns, use **glob** style matching.
-regexp	When matching string to the patterns, use regular expression matching (i.e., the same as implemented by the **regexp** command).
«—	Marks the end of options. The argument following this one will be treated

	as a string even if it starts with a -.

The **-regexp** option lets you match any regular expression, whereas **-glob** lets you use the shell filename matching methods. With **-glob**, the shell special characters *****, **[]**, **?** let you easily match on part of a string. With the **-regexp** option, you can match on complex alternative strings, specifying multiple instances of characters, the beginning or end of a string, and classes of characters. For example, to match on all filenames with the **.c** extension, you would use the following command:

```
switch -glob *.c
```

To match on words that end with a number, like "report17" and "report32," you could use the following command:

```
switch -regexp report[0-9]*
```

while

The **while** loop repeats commands. In Tcl, the **while** loop begins with the **while** command and takes two arguments, an expression, and either a single Tcl command or a block of Tcl commands. The expression is encased in braces. A block of Tcl commands begins with an opening brace on the same line as the **while** command. Then, on following lines are the Tcl commands that will be repeated in the loop. The block ends with a closing brace, usually on a line by itself. The syntax for the **while** loop with a single statement is described here:

```
 while {expression } {
 Tcl commands
 }
```

for

The **for** loop performs the same tasks as the **while** loop. However, it has a different format. The **for** loop takes four arguments, the first three of which are expressions and the last of which is a block of Tcl commands. The first three arguments are expressions that incorporate the initialization, test, and increment components of a loop. These expressions are each encased in braces. The last argument, the block of Tcl commands, begins with an opening brace and then continues with Tcl commands on the following lines, ending with a closing brace:

```
 for {expression1} {expression2} {expression3} {
 Tcl commands;
 }
```

foreach

The **foreach** structure is designed to sequentially reference a list of values. It is very similar to the C shell's **for-in** structure. The **foreach** structure takes three arguments: a variable, a list, and a block of Tcl commands. Each value in the list is assigned to the variable in the **foreach** structure. Like the **while** structure, the **foreach** structure is a loop. Each time through the loop, the next value in the list is assigned to the variable. When the end of the list is reached, the loop stops. As in the **while** loop, the block of Tcl commands is encased in braces. The syntax for the **foreach** loop is described here:

```
 foreach variable ( list of values ) {
 tcl commands
 }
```

Tcl Input and Output: gets and puts

Tcl can read input from the standard input or a file using the **gets** command and write output to the standard output with the **puts** command. The following command reads a line from the standard input, usually the keyboard. The input is placed in the variable line.

```
gets line
```

The **puts** command outputs a string to the standard output or to a file. It takes as its argument the string to be output.

```
puts $line.
```

gets reads a line into the variable specified as its argument. You can then use this variable to manipulate whatever has been read. For example, you can use **line** in a **puts** command to display what was input.

```
myread

#!/usr/bin/tclsh
gets line
puts "This is what I entered: $line"

$ myread
larisa and aleina
This is what I entered: larisa and aleina
```

You can use the **puts** command to write data to any file or to the standard output. File handle names are placed after the **puts** command and before any data such as strings or variables. If no filename is specified, then **puts** outputs to the standard output.

To output formatted values, you can use the results of a format command as the argument of a **puts** command. **format** performs a string conversion operation on a set of values, formatting them according to conversion specifiers in a format string.

```
puts [format "%s" $myname]
```

If you want to output different kinds of values in a **puts** operation, you can use the **format** command to first transform them into a single string. The **puts** command will only output a single string. In the following example, the contents of the **$firstname** and **$age** variables are output as a single string by first using the **format** command with two string specifiers, **"%s %d"**, to make them one string. **%d** will transform a numeric value into its corresponding character values.

```
puts [format "%s %d" $firstname $age]
```

For string values, you can just make them part of the single string by placing them within double quotes. In this example, **firstname** and **lastname** are made part of the same string:

```
puts "$firstname $lastname"
```

Tcl File Handles

You use the **open** command to create a file handle for a file or pipe (see Table 7 for a list of Tcl file commands). The **open** command takes two arguments, the filename and the file mode, and returns a file handle that can then be used to access the file. The filename argument can be the name of the file or a variable that holds the name of the file. The file mode is the permissions you are opening the file with. This can be **r** for read-only, **w** for write-only, and **a** for append only. To obtain both read and write permission for overwriting and updating a file, you attach a + to the file mode. **r+** gives you read and write permission. The syntax for **open** follows:

```
open ( filename-string, file-mode );
```

Table 7: Tcl: File Access and Input/Output Commands

File Access Commands	**Description**
file	Obtain file information
open	Open a file
close	Close a file
eof	Check for end of file
fcopy	Copy from one file to another
flush	Flush output from a file's internal buffers
glob	Match filenames using **glob** pattern characters
read	Read blocks of characters from a file
seek	Set the seek offset of a file
tell	Return the current offset of a file
socket	Open a TCP/IP network connection
Input/Output Commands	
format	Format a string with conversion specifiers, like **sprintf** in C
scan	Read and convert elements in a string using conversion specifiers, like **scanf** in C
gets	Read a line of input
puts	Write a string to output

You would usually use the **open** command in a **set** command so that you can assign the file handle returned by **open** to a variable. You can then use that file handle in that variable in other file commands to access the file. In the next example, the user opens the file **reports** with a file mode for reading, **r**, and assigns the returned file handle to the **myfile** variable.

```
set myfile [open "reports" r]
```

Often, the filename will be held in a variable. You then use the **$** with the variable name to reference the filename. In this example, the filename **reports** is held in the variable **filen**:

```
set myfile [open $filen r]
```

Once you have finished with the file, you close it with the **close** command. **close** takes as its argument the file handle of the file you want to close.

```
close $myfile
```

With the **gets** and **puts** commands, you can use a file handle to read and write from a specific file. **gets** takes two arguments: a file handle and a variable. It will read a line from the file referenced by the file handle and place it as a string in the variable. If no file handle is specified, then **gets** reads from the standard input. The following command reads a line from a file using the file handle in the **myfile** variable. The line is read into the **line** variable.

```
gets $myfile line
```

The **puts** command also takes two arguments: a file handle and a string. It will write the string to the file referenced by the file handle. If no file handle is specified, then **puts** will write to the standard output. In the following example, **puts** writes the string held in the **line** variable to the file referenced by the file handle held in **myfile**. Notice that there is a **$** before **line** in the **puts** command, but not in the previous **gets** command. **puts** operates on a string, whereas **gets** operates on a variable.

```
puts $myfile $line
```

```
myreport
```

```
#!/usr/bin/tclsh
 set reps [open "reports" r ]
 while ( gets $reps line)
 {
 puts $line;
 }
 close reps
```

You can use the **file** command to check certain features of files, such as whether they exist or if they are executable. You can also check for directories. The **file** command takes several options, depending on the action you want to take. The **exist** option checks whether a file exists, and the **size** option tells you its size. The isdirectory option determines whether the file is a directory, and **isfile** checks to see whether it is a file. With the **executable**, **readable**, and **writable** options, you can detect whether a file is executable, can be read, or can be written to. The **dirname** option displays the full pathname of the file, and the **extension** and **root** name options show the extension or root name of the file, respectively. The **atime**, **mtime**, and **owned** options display the last access time and the modification time, and whether it is owned by the user.

```
file exits reps
file isfile reps
file size reps
file executable myreport
```

Often filenames will be used as arguments to Tcl programs. In this case, you can use the **argv** list to obtain the filenames. The **argv** command lists all arguments entered on the command line when the Tcl script was invoked. You use the **lindex** command to extract a particular argument from the **argv** list. Many programs use filenames as their arguments. Many also specify options. Remember that the **lindex** command indexes a list from 0. So the first argument in the **argv** list would be obtained by the following (be sure to precede **argv** with the **$**):

```
lindex $argv 0
```

You can, if you wish, reference an argument in the **argv** list within the **open** command. Here, the **lindex** operation is enclosed in braces, in place of the filename. The **lindex** command will return the filename from the **argv** list.

```
set shandle [ open {lindex $argv 1} r ]
```

Tk

The Tk application extends Tcl with commands for creating and managing graphic objects such as windows, icons, buttons, and text fields. Tk commands create graphic objects using the X Window System. It is an easier way to program X Window objects than using the X11 Toolkit directly. With Tk, you can easily create sophisticated window-based user interfaces for your programs.

The Tk language is organized according to different types of graphic objects such as windows, buttons, menus, and scroll bars. Such objects are referred to as widgets. Each type of widget has its own command with which you can create a widget. For example, you can create a button with the button command or a window with the window command. A type of widget is considered a class, and the command to create such a widget is called a class command. The command will create a particular instance of that class, a particular widget of that type. button is the class command for creating a button. Graphical objects such as buttons and frames are also often referred to as widgets. Table 8 lists the different widgets available in Tk.

Table 8: Standard: TK Widgets

Widget	Description
button	A button
canvas	A window for drawing objects
checkbutton	A check button
entry	An input box
frame	A frame is a simple widget. Its primary purpose is to act as a spacer or container for complex window layouts

image
Create image objects for displaying pictures

label
A label

listbox
A list box with a selectable list of items

menu
A menu bar

menubutton
A menu button to access the menu

message
Create and manipulate message widgets

radiobutton
A radio button

scrollbar
A scroll bar

text
An editable text box

scale
A scale

Note Several currently available Tcl/Tk GUI builders are Free Visual Tcl, SpecTcl, VisualGIPSY, and XF. These are freely available, and you can download them from their Web sites or from the Tcl Developer Xchange

The wish Shell and Scripts

Tk operates under the X Window System. Within the X Window System, Tk uses its own shell, the wish shell, to execute Tk commands. To run Tk programs, you first start up your X-Window System and then start up the wish shell with the command wish. This will open up a window in which you can then run Tk commands.

You execute Tk commands within the wish shell interactively, entering commands and executing them one by one, or you can place the commands in a script file and execute them all at once. Usually, Tk commands are placed in a script that is then run with the invocation of the wish command. Like Tcl scripts, Tk scripts usually have the extension .tcl. For example, a Tk script called mydir.tcl would be read and executed by the following command entered in an Xterm window:

```
$ wish mydir.tcl
```

To create a standalone script that operates more like a command, you need to invoke the wish command within the script. Ordinarily the wish command will open an interactive Tk shell window whenever executed. To avoid this, you should invoke wish with the -f option.

```
#!/usr/bin/wish -f
```

Note When creating a standalone script, be sure to change its permissions with the chmod command to allow execution. You can then just enter the name of the script to run the program.

```
$ chmod 755 mydir1
$ ./mydir1
```

Tk Widgets

Tk programs consist of class commands that create various graphic widgets. The class command takes as its arguments the name you want to give the particular widget followed by configuration options with their values (see Table 9). Tk commands have a format similar to Tcl. You enter a Tk class command on a line, beginning with the name of the command followed by its arguments. Tk commands are more complicated than Tcl commands. Graphic interface commands require a significant amount of information about a widget to set it up. For example, a button requires a name, the text it will display, and the action it will take.

Table 9: Tk: Commands Event Operations

	Description
Bind	Associate Tcl scripts with X events
Bindtags	Bind commands to tags
Selection	Object or text selected by mouse
Geometry Managers	
Pack	Pack widgets next to each other
Place	Place widgets in positions in frame
Grid	Place widgets in a grid of rows and columns
Window Operations	
Destroy	Close a TK window

Toplevel
Select the top-level window

Wm
Set window features

Uplevel
Move up to previous window level

Many Tk commands can take various options indicating different features of a widget. Table 10 lists several options commonly used for Tk widgets. In the following example, a button is created using the button command. The button command takes as its first argument the name of the button widget. Then, different options define various features. The -text option is followed by a string that will be the text displayed by the button. The -command option is followed by the command that the button executes when it is clicked. This button command will display a button with the text "Click Me". When you click it, the Tk shell will exit.

```
button .mybutton -text "Click Me" -command exit
```

Table 10: Tk: Commonly Used Standard Options Button
Description

-activebackground
Specifies background color to use when drawing active elements

-activeborderwidth
Width of the 3-D border drawn around active elements

-activeforeground
Foreground color to use when drawing active elements

-anchor
How information is displayed in the widget; must be one of the values n, ne, e, se, s, sw, w, nw, or center

-background
The normal background color to use when displaying the widget

-font
The font to use when drawing text inside the widget

-foreground
The normal foreground color to use when displaying the widget

-geometry
Specifies the desired geometry for the widget's window

-image
Specifies an image to display in the widget

-insertbackground
Color to use as background in the area covered by the insertion cursor

-insertborderwidth
Width of the 3-D border to draw around the insertion cursor

-insertofftime
Number of milliseconds the insertion cursor should remain "off" in each blink cycle

-relief
Specifies the 3-D effect desired for the widget

-selectbackground
Specifies the background color to use when displaying selected items

-text
String to be displayed inside the widget

Button Options

-command
Specifies a Tcl command to associate with the button

-selectimage
Image to display when the check button is selected

-height
Height for the button

-state
Specifies one of three states for the radio button: normal, active, or disabled

-variable
Global variable to set to indicate whether or not this button is selected

-width
Width for the button

To set up a working interface, you need to define all the widgets you need to perform a given task. Some widgets are designed to manage other widgets; for instance, scroll bars are designed to manage windows. Other widgets, such as text input fields, may interact with a Tcl program. A menu choice may take the action of running part of a Tcl program.

Widgets are organized hierarchically. For example, to set up a window to input data, you may need a frame, within which may be text field widgets as well as buttons. Widget names reflect this hierarchy. The widget contained within another widget is prefixed with that widget's name. If the name of the frame is report and you want to call the text input field monday, the text input field will have the name report.monday. A period separates each level in the hierarchy. A button that you want to call ok that is within the report frame would be named report.ok.

Once you have created your widgets, their geometry has to be defined. The geometry determines the size of each widget in relation to the others, and where they are placed in the window. Tk has three geometry managers, pack, place, and grid. The pack command is used in these examples. When you have defined your widgets, you issue a geometry manager command on them to determine their size and shape on the screen.

Note Your widgets cannot be displayed until their geometry is determined.

The following determines the geometry of the .mybutton widget using the pack command:

```
pack .mybutton
```

The mydir1 program is a simple Tcl/Tk program to display a list of file and directory names in a Tk listbox widget with an attached scroll bar. Figure 1 shows this list box. With a listbox widget, you can display a list of items that you can then easily scroll through. Using the mouse, you can select a particular item. If a scroll bar is attached, you can scroll through the displayed items if there are more than can fit in the designated size of the list box. First the scroll bar is defined using the scrollbar command, giving it the name .scroll and binding it with the command .list yview. This instructs the scroll bar to be attached to the list box on a y-axis, vertical.

Figure 1: The mydir1 Tk list box

```
scrollbar .scroll -command ".list yview"
```

Then, the list box is defined with the listbox command, giving it the name .list and a y-axis scroll capability provided by the .scroll widget. The list box will appear sunken with the specified width and height.

```
listbox .list -yscroll ".scroll set" -relief sunken \
 -width 15 -height 15 -setgrid yes
```

The two widgets are then created with the pack command and positioned in the window. They are placed on the left side of the window and will always expand to fill the window. The anchor is on the west side of the window, w. The list box, .list, is placed first, followed by the scroll bar, .scroll.

```
pack .list .scroll -side left -fill both -expand yes -anchor w
```

A Tcl if test then follows that checks if the user entered an argument when the program was invoked. The if test checks to see if there is a first element in the argv list where any arguments are held. If there are no arguments, the current directory is used, as represented by the period. This chosen directory is assigned to the dir variable. A Tcl foreach operation is then used to fill the list box. The shell ls command, as executed with the exec command, obtains the list of files and directories. Each is then placed in the list box with the Tk insert operation for the .list widget. The insert command takes a position and a value. Here, the value is a filename held in $i that is placed at the end of the list.

```
.list insert end $i
```

The CTRL-C character is then bound to the exit command to allow you to easily close the window. A listing of the mydir1 program follows.

mydir1

```
#!/usr/bin/wish -f
# Create a scroll bar and listbox
scrollbar .scroll -command ".list yview"
listbox .list -yscroll ".scroll set" -relief sunken -width 15 -height 15 -setgrid yes
pack .list .scroll -side left -fill both -expand yes -anchor w
# If user enters a directory argument use that, otherwise use current directory.
if {$argc > 0} then {
 set dir [lindex $argv 0]
 } else {
 set dir "."
 }
# Fill the listbox (.list) with the list of files and directories obtained from ls
 cd $dir
 foreach i [exec ls -a ] {
 if [file isfile $i] {
 .list insert end $i
 }
 }
# Set up bindings for the file manager. Control-C closes the window.
bind all <Control-c> {destroy .}
```

To run the mydir1 program, first make it executable using the chmod command to set the executable permissions, as shown here:

```
chmod 755 mydir1
```

Then, within a terminal window on your desktop or window manager, just enter the mydir1 command at the prompt. You may have to precede it with a ./ to indicate the current directory.

```
./mydir1
```

A window will open with a list box displaying a list of files in your current working directory (see Figure 1). Use the scroll bar to display any filenames not shown. Click the window Close box to close the program.

Events and Bindings

A Tk program is event driven. Upon running, it waits for an event such as a mouse event or a keyboard event. A mouse event can be a mouse click or a double-click, or even a mouse down or up. A keyboard event can be a CTRL key or meta key, or even the ENTER key at the end of input data. When the program detects a particular event, it takes an action. The action may be another graphical operation such as displaying another menu, or it may be a Tcl, Perl, or shell program.

Bindings are the key operational component of a Tk program. Bindings detect the events that drive a Tk program. You can think of a Tk program as an infinite loop that continually scans for the occurrence of specified events (bindings). When it detects such an event, such as a mouse click or control key, it executes the actions bound to that event. These actions can be any Tcl/Tk command or series of commands. Usually, they call functions that can perform complex operations. When finished, the program resumes its scanning, looking for other bound events. This scanning continues indefinitely until it is forcibly broken by an exit or destroy command, as is done with the CTRL-C binding. You can think of bindings as multiple entry points where different parts of the program begin. It is not really the same structure as a traditional hierarchical sequential program. You should think of a binding as starting its own sequence of commands, its own program. This means that to trace the flow of control for a Tk program, you start with the bindings. Each binding has its own path, its own flow of control.

Actions are explicitly bound to given events using the bind command. The bind command takes as its arguments the name of a widget or class, the event to bind, and the action to bind to that event. Whenever the event takes place within that widget, the specified action is executed.

```
bind .myframe <CTRL-H> {.myframe delete insert }
```

You use the bind command to connect events in a Tk widget with the Tcl command you want executed. In a sense, you are dividing your Tcl program into segments, each of which is connected to an event in a Tk widget. When an event takes place in a Tk widget, its associated set of Tcl commands is executed. Other Tk commands, as well as Tcl commands, can be associated with an event bound to a Tk widget. This means that you can nest widgets and their events. The Tcl commands executed by one Tk event may, in turn, include other Tk commands and widgets with events bound to yet other Tcl commands.

Expect

Expect has several commands that you can use to automatically interact with any Unix program or utility that prompts you for responses. For example, the login procedure for different systems using FTP or telnet can be automatically programmed with Expect commands. Expect is designed to work with any interactive program. It waits for a response from a program and will then send the response specified in its script. You can drop out of the script with a simple command and interact with the program directly.

Three basic Expect commands are the send, expect, and interact commands. The expect command will wait to receive a string or value from the application you are interacting with. The send command will send a string to that application. The interact command places you into direct interaction with the application, ending the Expect/Tcl script. In the following script, Expect is used to perform an anonymous login with FTP. The spawn command starts up the FTP program. The Internet address of the FTP site is assumed to be an argument to this script, and as such will be held in the argv list. In place of $argv, you could put the name of a particular FTP site. The myftp.expect script that follows will set up an ftp connection automatically.

myftp.expect

```
#!/usr/bin/expect
spawn ftp
```

```
send "open $argv\r"
expect "Name"
send "anonymous\r"
expect "word:"
send "richlp@turtle.mytrek.com\r"
interact
```

To run Expect commands, you have to first enter the Expect shell. In the previous myftp.expect script, the Expect shell is invoked with the command #!/usr/bin/expect. Be sure to add execute permission with chmod 755 myftp.expect:

```
$myftp ftp.calderasystems.com
```

The expect command can take two arguments: the pattern to expect and an action to take if the pattern is matched. expect can also take as its argument a block of pattern/action arguments. In this case, expect can match on alternative patterns, executing the action only for the pattern it receives. For example, the ftp command may return a "connection refused" string instead of a "name" string. In that case, you would want to issue this message and exit the Expect script. If you want more than one action taken for a pattern, you can encase them in braces, separated by semicolons.

Another useful Expect command is timeout. You can set the timeout command to a number of seconds, then have Expect check for the timeout. To set the number of seconds for a timeout, you use set to assign it to the timeout variable (the default is 10 seconds). To have the expect command detect a timeout, you use the word timeout as the expect command's pattern. With the timeout, you can add an action to take. An example of an Expect script follows:

```
set timeout 20
end "open $argv\r"
expect {
 timeout {puts "Connection timed out\n"; exit }
 "Connection refused" {puts "Failed to connect\n"; exit}
 "Unknown host" {puts "$argv is unknown\n"; exit}
 "Name"
}
```

Expect can run with any kind of program that requires interaction. All you need to do is to determine the sequence of prompts and responses you want.

Gawk

Gawk is a programming language designed to let Linux users create their own shell filters. A filter operates within a Linux shell such as BASH or TCSH. It reads information from an input source such as a file or the standard input, modifies or analyzes that information, and then outputs the results. Results can be a modified version of the input or an analysis. For example, the sort filter reads a file and then outputs a sorted version of it, generating output that can be sorted alphabetically or numerically. The wc filter reads a file and then calculates the number of words and lines in it, outputting just that information. The grep filter will search a file for a particular pattern, outputting the lines the pattern is found on. With Gawk, you can design and create your own filters, in effect creating your own Linux commands. You can instruct Gawk to simply display lines of input text much like cat, or to search for patterns in a file like grep, or even count words in a file like wc. In each case, you could add your own customized filtering capabilities. You could display only part of each line, or search for a pattern in a specific field, or count only words that are capitalized. This flexibility lets you use Gawk to generate reports, detecting patterns and performing calculations on the data.

You can use Gawk directly on the shell command line, or you can place Gawk within a shell file that you can then execute. The name of the shell file can be thought of as a new filter that you have created. In effect, with Gawk, you can define your own filters. In this sense there are two ways of thinking about Gawk. Gawk is itself a filter that you can invoke on the command line like any other filter, and Gawk is a programmable filter that you can use to create your own filters. This section will examine both aspects of Gawk. First we will examine Gawk as a filter, with all its different features. Then, we will see how you can use Gawk to define your own filters.

The Gawk utility has all the flexibility and complexity of a programming language. Gawk has a set of operators that allow it to make decisions and calculations. You can also declare variables and use them in control structures to control how lines are to be processed. Many of the programming features are taken from the C programming language and share the same syntax. All of this makes for a very powerful programming tool.

Gawk is the GNU version of the Unix Awk utility. Awk was originally created as a standard utility for the Unix operating system. One of its creators is Brian Kernighan, who developed the Unix operations system. An enhanced version of Awk called Nawk was developed later to include file handling. With Nawk, you can access several files in the same program. Gawk is a further enhancement, including the added features of Nawk as well as the standard capabilities of Awk.

Gawk has a full set of arithmetic operators. You can perform multiplication, division, addition, subtraction, and modulo calculations. The arithmetic operators are the same as those used in the C programming language and Perl. Gawk also supports both scalar and associative arrays. Gawk has control structures similar to those in the C programming language, as well as pattern matching and string functions similar to those in Perl and Tcl/Tk. You can find out more about Gawk at www.gnu.org/software/gawk.

The gawk Command

The gawk command takes as its arguments a Gawk instruction and a list of filenames. The Gawk instruction is encased in single quotes and is read as one argument. The Gawk instruction itself consists of two segments: a pattern and an action. The action is enclosed in brackets. The term "pattern" can be misleading. It is perhaps clearer to think of the pattern segment as a condition. The pattern segment can be either a pattern search or a test condition of the type found in programming languages. The Gawk utility has a full set of operators with which to construct complex conditions. You can think of a pattern search as just one other kind of condition for retrieving records. Instead of simply matching patterns as in the case of grep, the user specifies a condition. Records that meet that condition are then retrieved. The actions in the action segment are then applied to the record. The next example shows the syntax of a Gawk instruction, which you can think of as condition {action}:

```
pattern {action}
```

The Gawk utility operates on either files or the standard input. You can list filenames on the command line after the instruction. If there are no filenames listed, input is taken from the standard input. The example below shows the structure of the entire Gawk instruction. The invocation of Gawk consists of the gawk keyword followed by a Gawk instruction and filenames. As with the sed commands, the instruction should be placed within single quotes to avoid interpretation by the shell. Since the condition and action are not separate arguments for Gawk, you need to enclose them both in one set of quotes. The next example shows the syntax of a gawk command:

```
$ gawk 'pattern action { }' filenames
```

You can think of the pattern in a Gawk instruction as referencing a line. The Gawk action is then performed on that line. The next two examples below print all lines with the pattern "Penguin". The pattern segment is a pattern search. A pattern search is denoted by a pattern enclosed in slashes. All records with this pattern are retrieved. The action segment in the first example contains the print command. The print command outputs the line to the standard output.

books

```
Tempest Shakespeare 15.75 Penguin
Christmas Dickens 3.50 Academic
Iliad Homer 10.25 Random
Raven Poe 2.50 Penguin
```

```
$ gawk '/Penguin/{print}' books
Tempest Shakespeare 15.75 Penguin
Raven Poe 2.50 Penguin
```

Tip Both the action and pattern have defaults that allow you to leave either of them out. The print action is the default action. If an action is not specified, the line is printed. The default pattern is the selection of every line in the text. If the pattern is not specified, the action is applied to all lines.

In the second example, there is no action segment. The default action is then used, the print action.

```
$ gawk '/Penguin/' books
Tempest Shakespeare 15.75 Penguin
Raven Poe 2.50 Penguin
```

Pattern Searches and Special Characters

Gawk can retrieve lines using a pattern search that contains special characters. The pattern is designated with a beginning and ending slash, and placed in the pattern segment of the Gawk instruction.

```
/pattern/ {action}
```

The pattern search is performed on all the lines in the file. If the pattern is found, the action is performed on the line. In this respect, Gawk performs very much like an editing operation. Like sed, a line is treated as a line of text and the pattern is searched for throughout the line. In the next example, Gawk searches for any line with the pattern "Poe". When a match is found, the line is output.

```
$ gawk '/Poe/{print}' books
Raven Poe 2.50 Penguin
```

You can use the same special characters for Gawk that are used for regular expressions in the sed filter and the Ed editor. The first example below searches for a pattern at the beginning of the line. The special character ^ references the beginning of a line. The second example searches for a pattern at the end of a line using the special character $:

```
$ gawk '/^Christmas/{print}' books
Christmas Dickens 3.50 Academic

$ gawk '/Random$/{print}' books
Iliad Homer 10.25 Random
```

As in Ed and sed, you can use special characters to specify variations on a pattern. The period matches any character, the asterisk matches repeated characters, and the brackets match a class of characters: ., *, and []. In the first example below, the period is used to match any pattern in which a single character is followed by the characters "en":

```
$ gawk '/.en/{print}' books
Tempest Shakespeare 15.75 Penguin
Christmas Dickens 3.50 Academic
Raven Poe 2.50 Penguin
```

The next example uses the brackets and asterisk special characters to specify a sequence of numbers. The set of possible numbers is represented by the brackets enclosing the range of numeric characters [0–9]. The asterisk then specifies any repeated sequence of numbers. The context for such a sequence consists of the characters ".50". Any number ending with .50 will be matched. Notice that the period is quoted with a backslash to treat it as the period character, not as a special character.

```
$ gawk '/[0-9]*\.50/ {print}' books
Christmas Dickens 3.50 Academic
Raven Poe 2.50 Penguin
```

Gawk also uses the extended special characters: +, ?, and |. The + and ? are variations on the * special character. The + matches one or more repeated instances of a character. The ? matches zero or one instance of a character. The | provides alternative patterns to be searched. In the next example, the user searches for a line containing either the pattern "Penguin" or the pattern "Academic":

```
$ gawk '/Penguin|Academic/ {print}' books
Tempest Shakespeare 15.75 Penguin
Christmas Dickens 3.50 Academic
Raven Poe 2.50 Penguin
```

Variables

Gawk provides for the definition of variables and arrays. It also supports the standard kind of arithmetic and assignment operators found in most programming languages such as C. Relational operators are also supported.

In Gawk, there are three types of variables: field variables, special Gawk variables, and user-defined variables. Gawk automatically defines both the field and special variables. The user can define his or her own variables. You can also define arithmetic and string constants. Arithmetic constants consist of numeric characters, and string constants consist of any characters enclosed within double quotes.

Field variables are designed to reference fields in a line. A field is any set of characters separated by a field delimiter. The default delimiter is a space or tab. As with other database filters, Gawk numbers fields from 1. This is similar to the number used for arguments in shell scripts. Gawk defines a field variable for each field in the file. A field variable consists of a dollar sign followed by the number of the field. $2 references the second field. The variable $0 is a special field variable that contains the entire line.

Tip A variable may be used in either the pattern or action segment of the Gawk instruction. If more than one variable is listed, they are separated by commas. Notice that the dollar sign is used differently in Gawk than in the shell.

In the next example, the second and fourth fields of the books file are printed out. The $2 and $4 reference the second and fourth fields.

books

```
Tempest Shakespeare 15.75 Penguin
Christmas Dickens 3.50 Academic
Iliad Homer 10.25 Random
Raven Poe 2.50 Penguin
```

```
$ gawk '{print $2, $4}' books
Shakespeare Penguin
Dickens Academic
Homer Random
Poe Penguin
```

In the next example, the user outputs the line with the pattern "Dickens" twice—first reversing the order of the fields and then with the fields in order. The $0 is used to output all the fields in order, the entire line.

```
$ gawk '/Dickens/ {print $4, $3, $2, $1; print $0}' books
Academic 3.50 Dickens Christmas
Christmas Dickens 3.50 Academic
```

Gawk defines a set of special variables that provide information about the line being processed. The variable NR contains the number of the current line (or record). The variable

NF contains the number of fields in the current line. There are other special variables that hold the field and record delimiters. There is even one, FILENAME, that holds the name of the input file. The Gawk special variables are listed in Table 11.

Table 11: Gawk: Special Variables

Variables	Description
NR	Record number of current record
NF	Number of fields in current record
$0	The entire current record
$n	The fields in the current record, numbered from 1—for example, $1
FS	Input field delimiter; default delimiter is space or tab
FILENAME	Name of current input file

Both special variables and user-defined variables do not have a dollar sign placed before them. To use such variables, you only need to specify their name. The next example combines both the special variable NR with the field variables $2 and $4 to print out the line number of the line followed by the contents of fields two and four. The NR variable holds the line number of the line being processed.

```
$ gawk '{print NR, $2, $4}' books
1 Shakespeare Penguin
2 Dickens Academic
3 Homer Random
4 Poe Penguin
```

You can also define your own variables, giving them any name you want. Variables can be named using any alphabetic or numeric characters as well as underscores. The name must begin with an alphabetic character. A variable is defined when you first use it. The type of variable is determined by the way it is used. If you use it to hold numeric values, the variable is considered arithmetic. If you use it to hold characters, the variable is considered a string. You need to be consistent in the way in which you use a variable. String variables should not be used in arithmetic calculations and vice versa.

You assign a value to a variable using the assignment operator, =. The left-hand side of an assignment operation is always a variable and the right-hand side is the value assigned to it. A value can be the contents of a variable such as a field, special, or other user variable. It can also be a constant. In the next example, the user assigns the contents of the second field to the variable myfield:

```
$ gawk '{myfield = $2; print myfield}' books
Shakespeare
Dickens
Homer
Poe
```

By default, Gawk separates fields by spaces or tabs. However, if you want to use a specific delimiter, you need to specify it. The -F option allows Gawk to detect a specific delimiter. The -F option actually sets a Gawk special variable called FS, which stands for field separator. With the -F option, you can use any character you want for your delimiter.