Android Architecture For Beginners

Leon Romanovsky
leon@leon.nu

www.leon.nu

April 22, 2013

Introduction

Linux-based operating system with market share - 69.70% in smartphones, 42% in tablets, available on smart TVs and mini PC.

History

- October 2003 - Android Inc. founded by Andy Rubin, Rich Miner, Nick Sears and Chris White
- August 2005 - Google acquired Android Inc.
- November 2007 - Open Handset Alliance (OHA) formed
- September 2008 - Android 1.0 released

Android Stack

Linux Kernel Layer

HAL, memory management, security, power management, drivers and network

Native Libraries

core libraries to support different types of data (audio and video formats, e.t.c.), Java libraries (Dalvik VM)

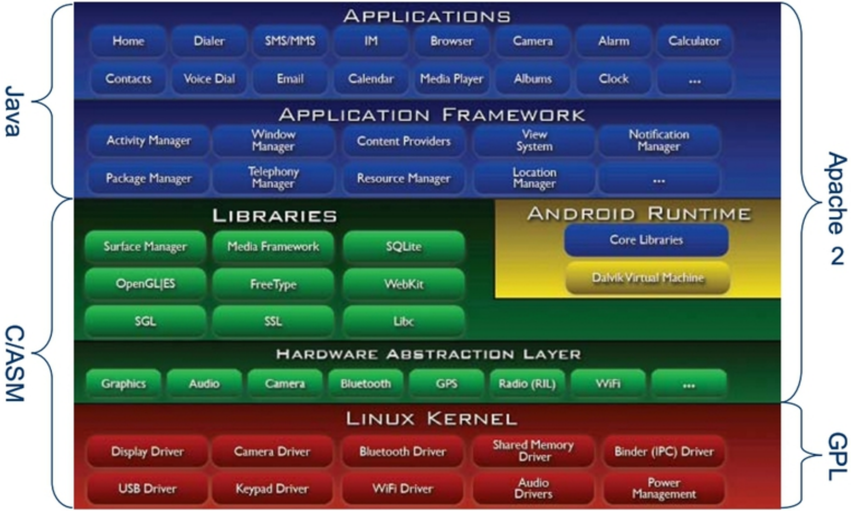
Application Framework

basic functions of device (in use by developers)

Application

system installed (/system/app) and user installed (/data/app)

Layers Diagram



Java

C/ASM

Apache 2

GPL

APPLICATIONS

Home, Dialer, SMS/MMS, IM, Browser, Camera, Alarm, Calculator

Contacts, Voice Dial, Email, Calendar, Media Player, Albums, Clock, ...

APPLICATION FRAMEWORK

Activity Manager, Window Manager, Content Providers, View System, Notification Manager

Package Manager, Telephony Manager, Resource Manager, Location Manager, ...

LIBRARIES

Surface Manager, Media Framework, SQLite

OpenGL|ES, FreeType, WebKit

SGL, SSL, Libc

ANDROID RUNTIME

Core Libraries

Dalvik Virtual Machine

HARDWARE ABSTRACTION LAYER

Graphics, Audio, Camera, Bluetooth, GPS, Radio (RIL), WiFi, ...

LINUX KERNEL

Display Driver, Camera Driver, Bluetooth Driver, Shared Memory Driver, Binder (IPC) Driver

USB Driver, Keypad Driver, WiFi Driver, Audio Drivers, Power Management

Linux Kernel vs. Android Kernel

core code

- binder - interprocess communication mechanism
- ashmem - shared memory mechanism
- logger
- timestamps

performance/power

- wakelock
- low-memory killer
- CPU frequency governor

and much more . . . 361 Android patches for the kernel

Security Architecture

- Based on Linux kernel
    - A user-based permissions model (user/group ID)
    - Process isolation (sandboxing)
    - Extensible mechanism for secure IPC
- Mandatory application sandbox for all applications
- Secure interprocess communication
    - ContentProviders, Intents, Binder/IPC, local sockets, or the file system
- Application signing
    - Based on Java's JAR specification
- Application-defined and user-granted permissions
    - Apps statically declare permissions they need (use)
    - No support for dynamic (run-time) granting of permissions

Special Permissions

frameworks/base/data/etc/platform.xml

```
<permissions>
   <permission name=“android.permission.CAMERA” >
      <group gid=“camera” />
   </permission>
   <permission name=“android.permission.BLUETOOTH” >
      <group gid=“net_bt” />
   </permission>
   ...
</permissions>
```

Android Directory Structure

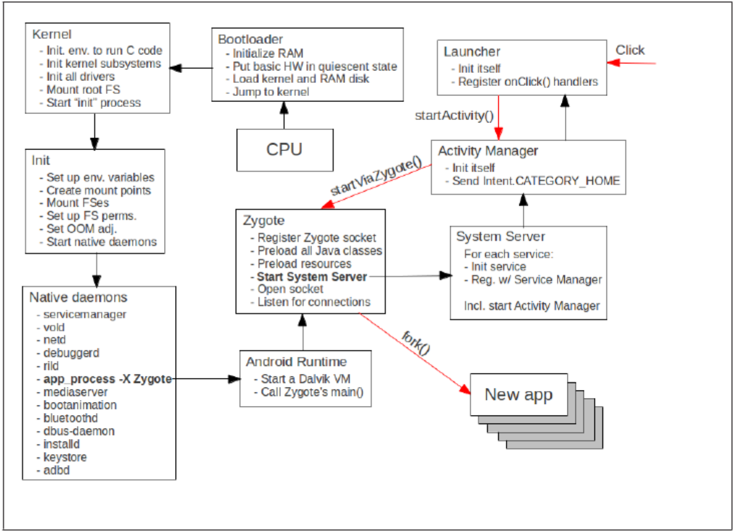
Based on Linux

- /dev, /proc, /sys, /mnt, /etc

Android specific

- /system - read-only main Android directory
- /data - read-write local storage (/data/app, /data/data)
- /cache - temporary storage

Boot Process



Kernel
- Init. env. to run C code
- Init kernel subsystems
- Init all drivers
- Mount root FS
- Start "init" process

Bootloader
- Initialize RAM
- Put basic HW in quiescent state
- Load kernel and RAM disk
- Jump to kernel

CPU

Launcher
- Init itself
- Register onClick() handlers

Init
- Set up env. variables
- Create mount points
- Mount FSes
- Set up FS perms.
- Set OOM adj.
- Start native daemons

Activity Manager
- Init itself
- Send Intent.CATEGORY_HOME

Zygote
- Register Zygote socket
- Preload all Java classes
- Preload resources
- **Start System Server**
- Open socket
- Listen for connections

System Server

For each service:
- Init service
- Reg. w/ Service Manager

Incl. start Activity Manager

Native daemons
- servicemanager
- vold
- netd
- debuggerd
- rild
- **app_process -X Zygote**
- mediaserver
- bootanimation
- bluetoothd
- dbus-daemon
- installd
- keystore
- adbd

Android Runtime
- Start a Dalvik VM
- Call Zygote's main()

New app

SoC Boot For Pedants

Raspberry Pi

A small RISC core on the GPU (VideoCore IV©Multimedia Co-Processor) is responsible for booting the SoC.

Qualcomm

There are two processors in the MSM 7x30, an ARM9 for the radio and an ARM11 auxiliary applications processor. The ARM9 running REX loads the eMMC hboot partition into memory at 0x8D00000 (virtual) and starts the ARM11 auxiliary applications processor executing at this location.

Nvidia Tegra

A co-processor known as the AVP (Audio-Video Processor). This processor implements the initial boot process, it is an ARM7TDMI. When Tegra is powered on, the AVP executes code from the boot ROM.

init

system/core/init/init.c

```
int main(int argc, char **argv)
{ . . .
  /* clear the umask */
  umask(0);
  /* Get the basic filesystem setup we need put
   * together in the initramdisk on / and then we will
   * let the rc file figure out the rest.
   */
  mkdir(“/dev”, 0755);
  mkdir(“/proc”, 0755);
  mkdir(“/sys”, 0755);
  mount(“tmpfs”, “/dev”, “tmpfs”, MS_NOSUID, “mode=0755”);
  . . .
  init_parse_config_file(“/init.rc”);
  . . .
}
```

init.rc

action - trigger

system/core/rootdir/init.rc

```
import /init.usb.rc
import /init.${ro.hardware}.rc
import /init.trace.rc
...
on fs
# mount mtd partitions
  # Mount /system rw first to give the filesystem a
  ...chance to save a checkpoint
  mount yaffs2 mtdsystem /system
  mount yaffs2 mtd@system /system ro remount
...
```

Further Reading

- Embedded Android by Karim Yaghmour
- Android Security Underpinnings by Marko Gargenta
- Working with MTD Devices
- Understanding Android Security by William Enck et al.
- Android Security Overview