Android: Under the Hood

•••

GDG-SG DevFest 5th Nov 2016

Jason Zaman

Overview

- Who am I?
- Android Block Diagram
- Mobile Hardware
- Filesystem Layout
- Startup
- Linux Kernel
- Bionic libc
- Ashmem / Binder IPC
- Zygote
- Dalvik VM
- ART

$ whoami

Jason “perfinion” Zaman

jason@perfinion.com GPG keyid: 0x7EF137EC935B0EAF

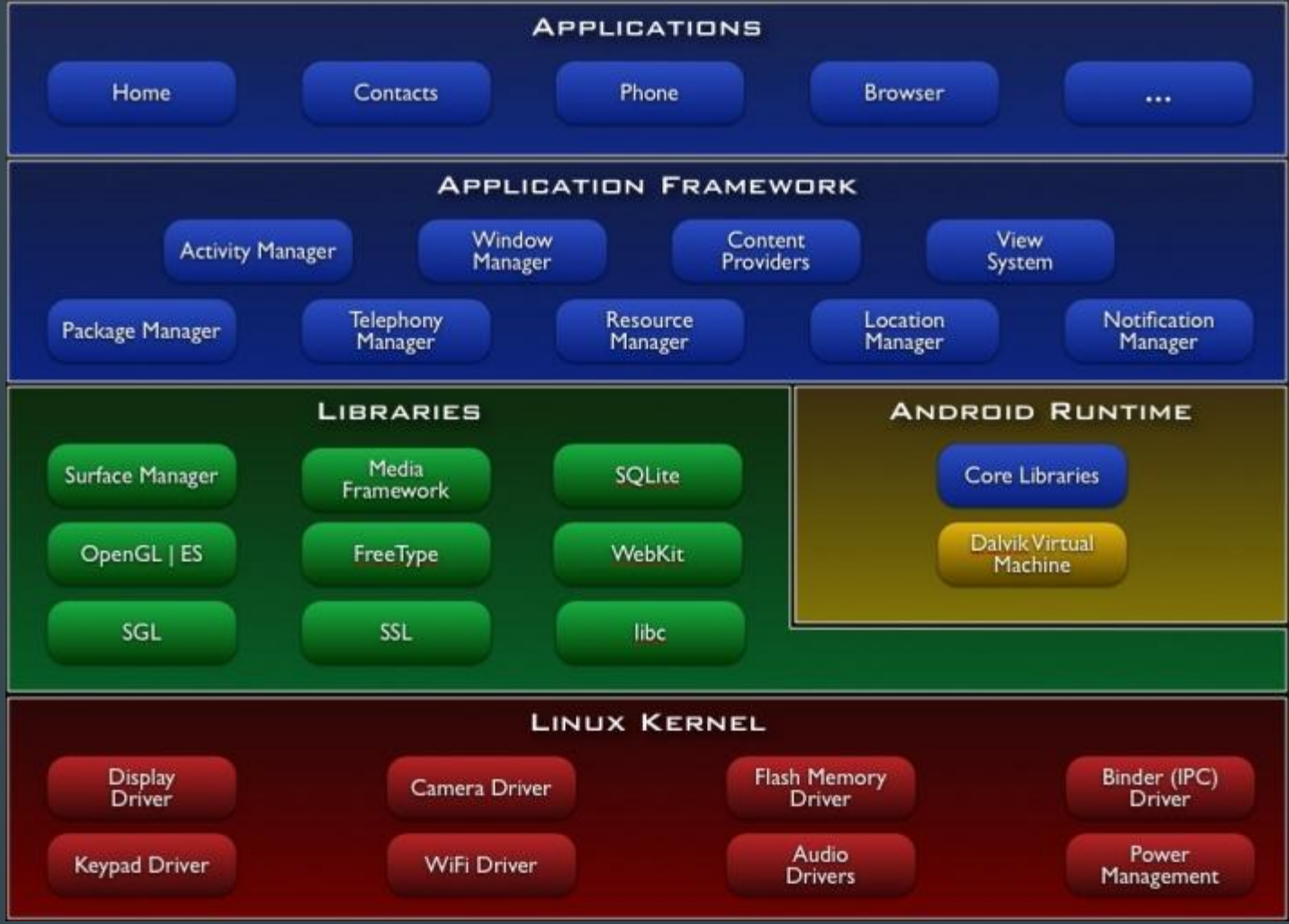
Blog: http://blog.perfinion.com/

Twitter: @perfinion Github: github.com/perfinion

Built Android apps with >7MM users, Used Android since 1.0 / G1

Gentoo Linux developer, SELinux and Hardened projects

Android Block Diagram

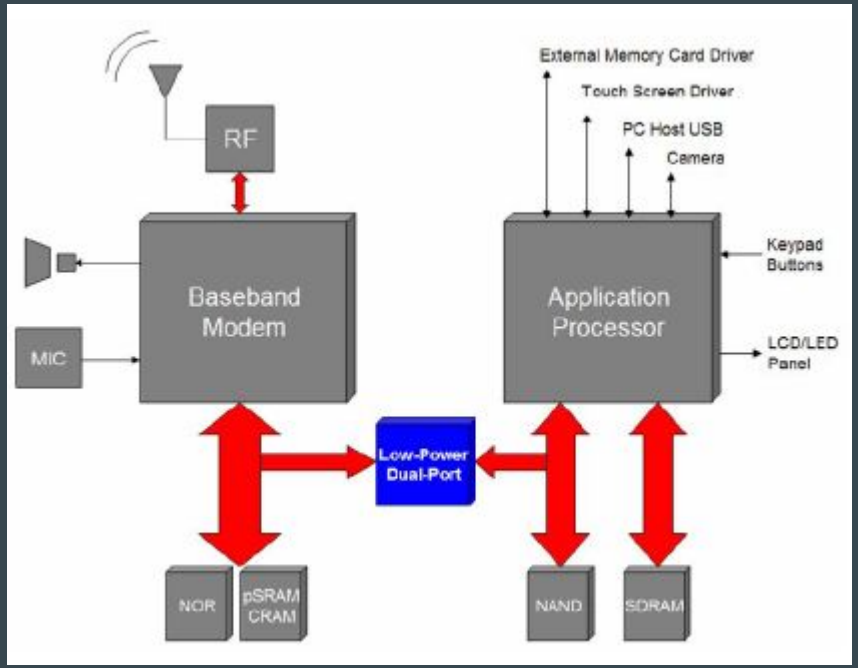


From: https://developer.android.com/images/system-architecture.jpg

Mobile Hardware

Mobile processors are different than desktop

- Baseband (radio) processor
  - Made by Qualcomm / MediaTek / whoever
  - Not open source
  - Handles GSM/3G/LTE communication
  - Low power
- Application processor
  - Android runs here
  - Linux
  - More powerful
- App CPU can shut down completely to save power



From: http://www.eetimes.com/document.asp?doc_id=1274981

Filesystem Layout

- / - initramfs, read-only ramdisk
  - Init lives here
- /system - read-only, ext4 / yaffs
- /data - read-write, ext4 / yaffs
  - Non-app data
- /data/data - read-write, ext4 / yaffs
  - Stores all your app data
- /sdcard - FAT32
  - Sometimes real sdcard
  - Sometimes emulated from /data partition
- /cache - system cache, ext4 / yaffs
  - Firmware updates
  - Play Store temp downloads

Startup

1. Power button is pressed
2. Baseband starts up, initializes itself
3. Starts App processor, loads bootloader
4. Loads Linux kernel
5. Initializes device drivers
6. Mounts initramfs on /, executes /init
7. Reads /init.rc
8. Shows boot splash
9. Mounts all the partitions
10. Set FS permissions
11. Starts system daemons
    - Adbd, rild, netd, vold, logd, servicemanager
12. Starts Zygote
    - All Java based stuff comes from here
13. Starts SystemServer
14. Starts other system services / managers
    - Activity Manager
    - Package Manager
    - Window Manager
    - Location Manager
    - …

Linux Kernel

- Supports lots of hardware
- Scheduler
- Task separation
- UID/GIDs
  - Each app is sandboxed in a separate Linux user
  - adb shell ls -al /data/data/
  - u0_a17 in ps
- SELinux
  - MCS categories to confine apps even more
- Verified Boot
  - dm-verity
- Frequently custom forks
  - Eg. Qualcomm MSM kernel
- Some new features added for Android
  - Low-mem killer vs OOM killer
  - Binder IPC
  - Ashmem
  - Wake Locks / Opportunistic Sleep
- Other features disabled
  - Most other IPC is disabled and not supported
  - Tons of other parts

Bionic libc

- Android's C Library
  - Largely based off BSD's libc
  - Designed for embedded
- *Much* smaller
  - GLibc: ~2MB
  - Bionic: ~200kB
- Adds Android-specific features
  - Logcat

- Not compatible with GNU Glibc
- Must recompile native libs for android, cannot use regular linux as-is.
- Removes lots of features
  - No Unicode
    - Use proper unicode libs
  - No SYSV IPC
    - Disallowed completely, only Binder
  - No Locales

Ashmem

Android shared memory

- Similar to POSIX SHM
- Designed for low memory systems
  - Allows for memory reclaim
- Reclaimed on process exit
- Shared with names vs before fork()
- Used to implement Binder

Binder IPC

- InterProcess-Communication used in Android
- Similar to CORBA
- Originally designed long before Android
- Now merged into mainline linux kernel too.
- Android uses this for all IPC
  - Parcelable
  - Intents
  - Services
  - context.getSystemService(NOTIFICATION_SERVICE);

Zygote

noun, Biology.

1. the cell produced by the union of two gametes, before it undergoes cleavage

- Every Android app is launched from here
- Started very early during boot
- Pre-loads a lot of libraries to save time / memory later
- Listens on socket (/dev/socket/zygote) for commands to launch new apps

```
startActivity(intent)
⇒ Binder
⇒ ActivityManager.startActivity(...);
⇒ Process.startViaZygote(...);
⇒ sends command via socket
⇒ zygote process
⇒ fork();
⇒ Load apk
⇒ Start main thread Looper
⇒ Initialize Application
⇒ Initialize MainActivity
```

Dalvik VM

Pre-Lollipop: Dalvik VM
Own bytecode optimized for mobile
Low RAM / No Swap / Tiny CPU

Compile:

- .java ⇒ javac ⇒ .class ⇒ dx ⇒ classes.dex

Install:

- classes.dex ⇒ dexopt ⇒ Verified and optimized .odex
- Not compiled

Run:

- Load .apk, load classes.dex
- Interpret dx code
- JIT the hot paths

More info:
https://android.googlesource.com/platform/dalvik/+/lollipop-release/docs/dexopt.html

ART - Android Run-Time

- New in Lollipop
- AOT (Ahead of Time) vs JIT (Just In Time)
- New Garbage Collector (GC)

Compile:

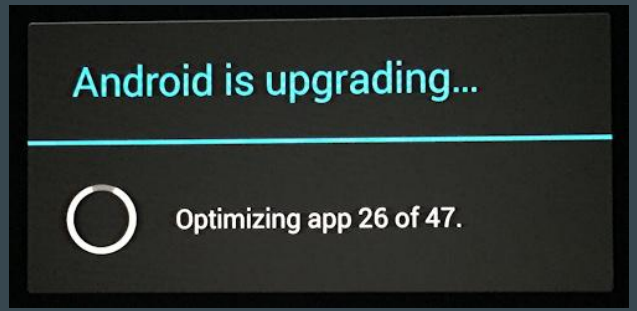
- .java ⇒ javac ⇒ .class ⇒ dx ⇒ classes.dex
- Exactly the same; no change to apk

Install:

- classes.dex ⇒ art ⇒ native executable specifically for processor

Run:

- Load app
- Run it



From: http://cdn-3.askdavetaylor.com/wp-content/uploads/2015/02/android-phone-system-update-8.jpg

ART - Android Run-Time in Android Nougat 7.0

Now both AOT + JIT

Compile:

- Still the same

Install:

- Classes.dex ⇒ art ⇒ native compilation for parts
- Saves disk space
- Saves time

Run:

- Run native bits
- Interpret uncompiled bits
- Save profiling data for later

When idle and charging:

- Go through all apps
- Examine profiling data
- Re-compile app using new profile

More info: The Evolution of ART - Google I/O 2016