Android Deep Dive With Dr. Chuck Easttom

About the instructor

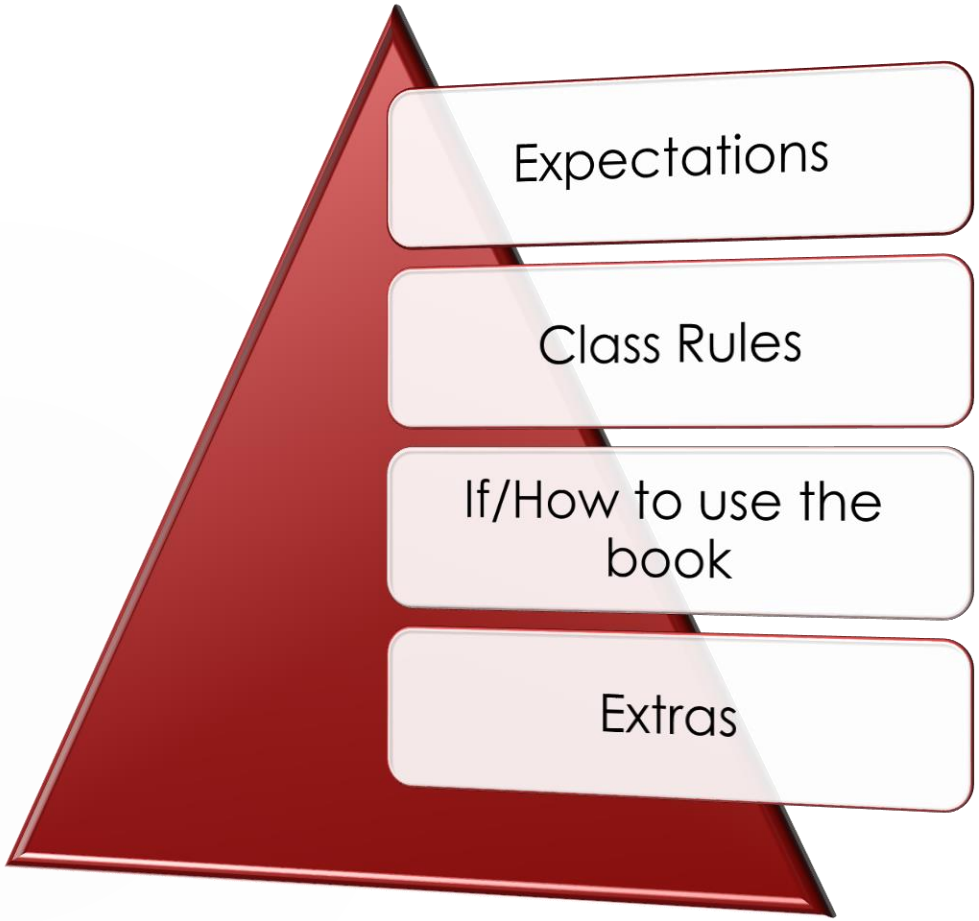
www.chuckeasttom.com

chuckeasttom@gmail.com

Or LinkedIn

https://www.linkedin.com/in/chuck-easttom-m-ed-mba-msse-d-sc-482568184/

The Class



Goals

- Deep understanding of the Android Architecture and operating system
- Be able to use ADB effectively
- Use the Android as a hacking platform
- Hack into an Android phone and take control
- Perform essential forensics on an Android
- Understand forensic science
- Have a basic understanding of Android programming
- Lots of HANDS ON
- As many extras as we can!

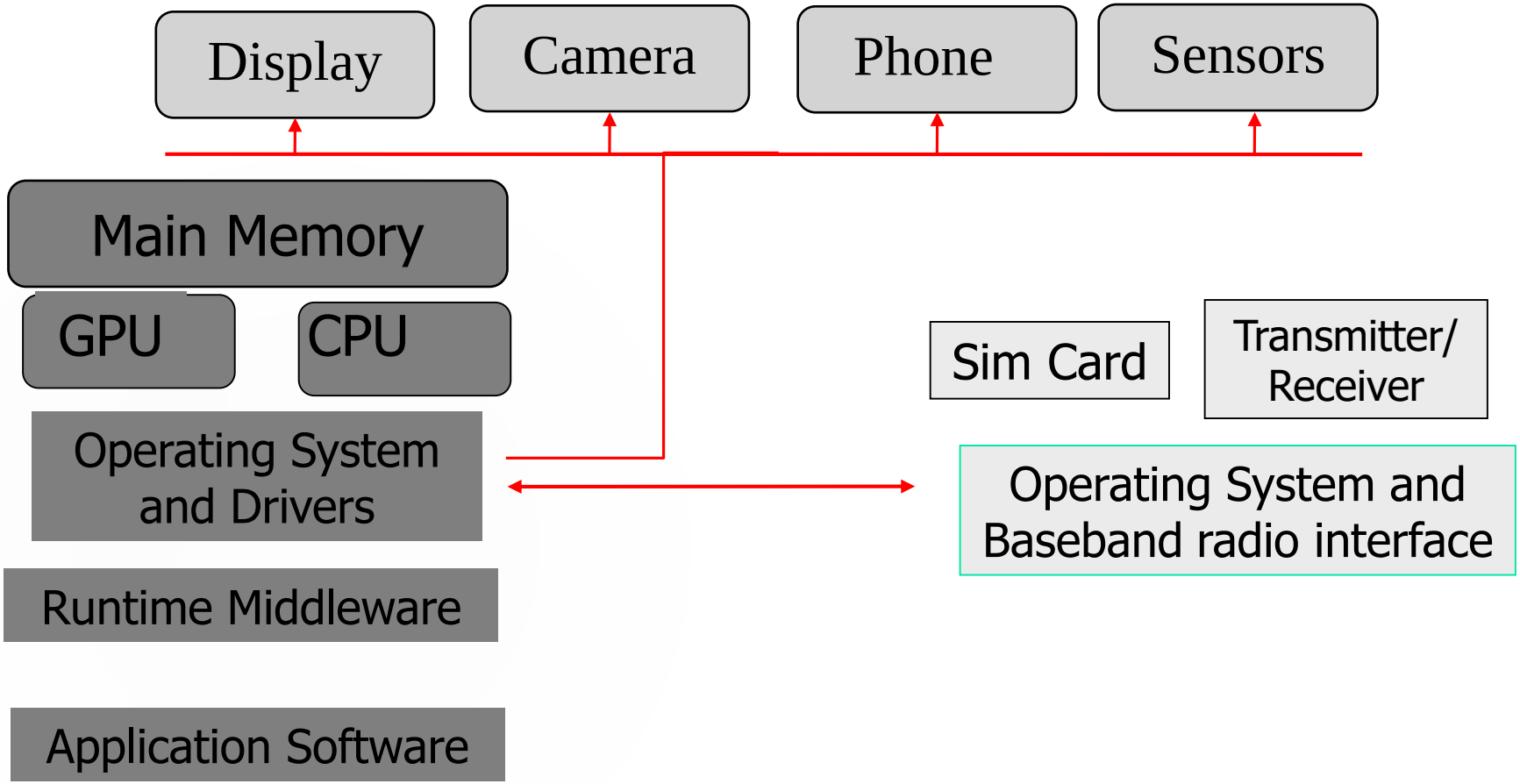
Some Facts

- Currently Android is 86.1% of the market for mobile devices.
- Apple iOS is 13.7% of the market for mobile devices.
- Default browser is Google Chrome, but Chrome is based on the Blink browser engine.

Zero Day for Android

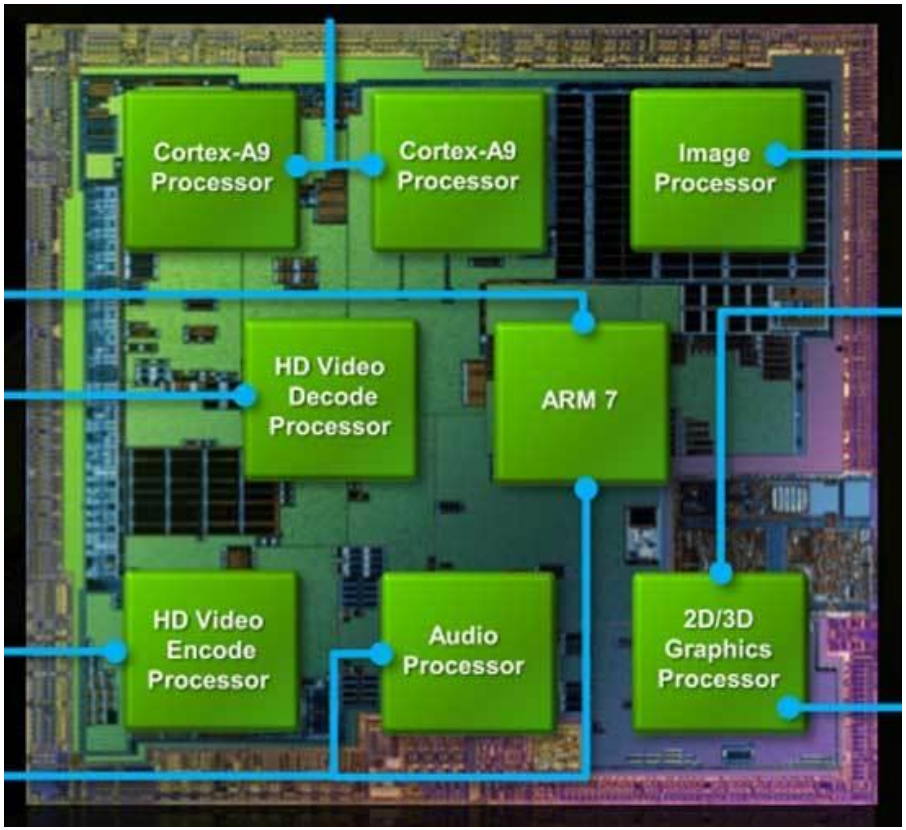
- September 4, 2019
- The vulnerability resides in how the Video for Linux (V4L2) driver that's included with the Android OS handles input data.
- Feeding the driver malicious input can allow an attacker to elevate their access from a lowly user to root access.
- One scenario where this zero-day can come in handy is when malware authors bundle it within malicious apps they distribute via the official Play Store or through third-party app stores.
- After the user installs one of these malicious apps, the zero-day can grant the malicious app root access, and the app can then carry out any operations it wants -- stealing user data, downloading other apps, etc..
- https://www.zdnet.com/article/zero-day-disclosed-in-android-os/

Hardware Architecture



Central Processing Unit (CPU)

- ARM architecture (majority)
- System-on-a-Chip
- 32-bit or 64-bit.
- Low power consumption
- Multi-core



Instruction Set Design Decisions

- Operation repertoire
  - How many ops?
  - What can they do?
  - How complex are they?
- Data types – various types of operations and how they are performed
- Instruction formats
  - Length of op code field
  - Number of addresses

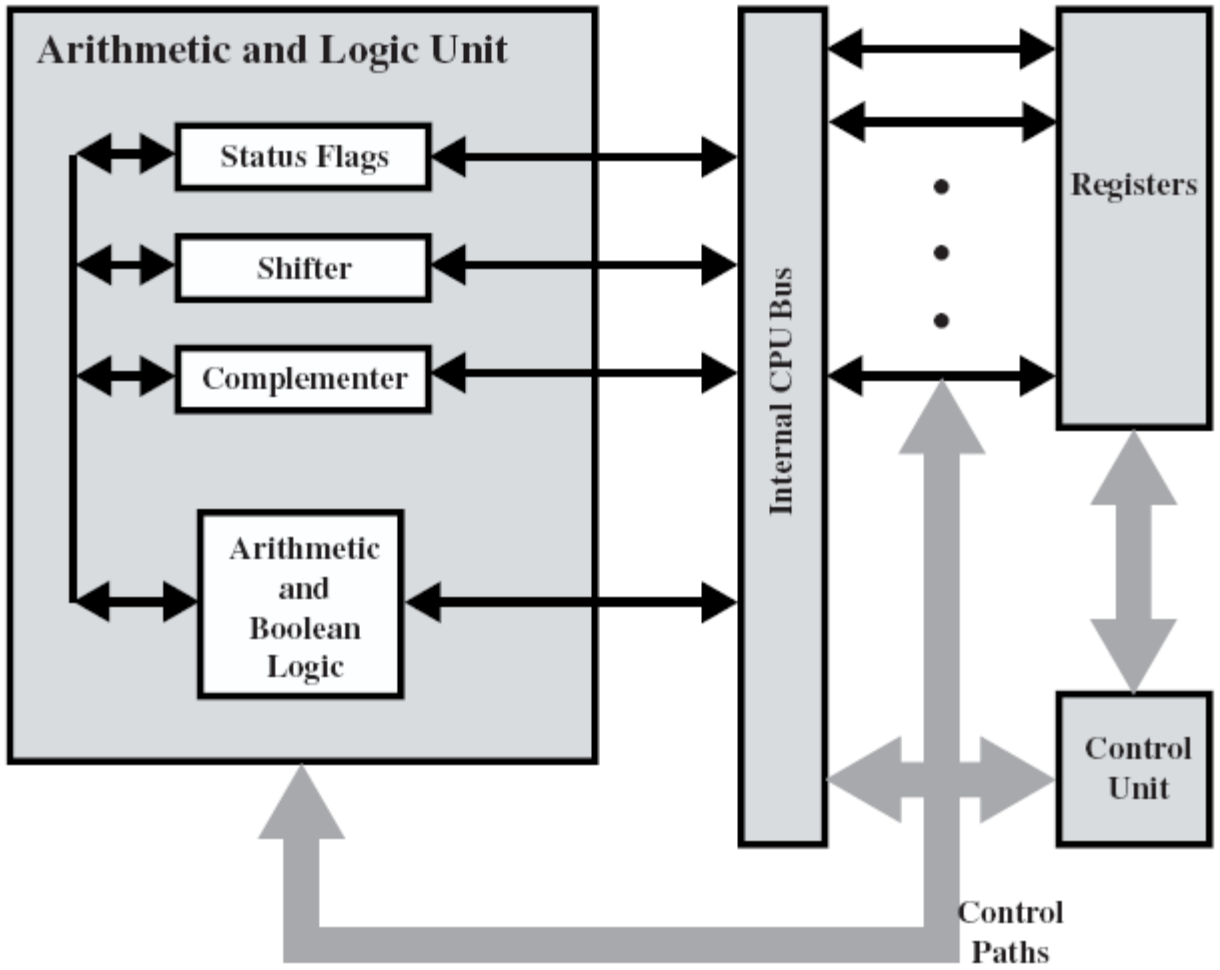
CPU Internal Design Issues

From our discussion of the architecture of the computer, we've put some requirements on the CPU.

- CPU fetches instructions from memory
- CPU interprets instructions to determine action that is required
- CPU fetches data that may be required for execution (could come from memory or I/O)
- CPU processes data with arithmetic, logic, or some movement of data
- CPU writes data (results) to memory or I/O

CPU Internal Structure

Design decisions here affect instruction set design



CPU Internal Structure – ALU, Internal CPU Bus, and Control Unit

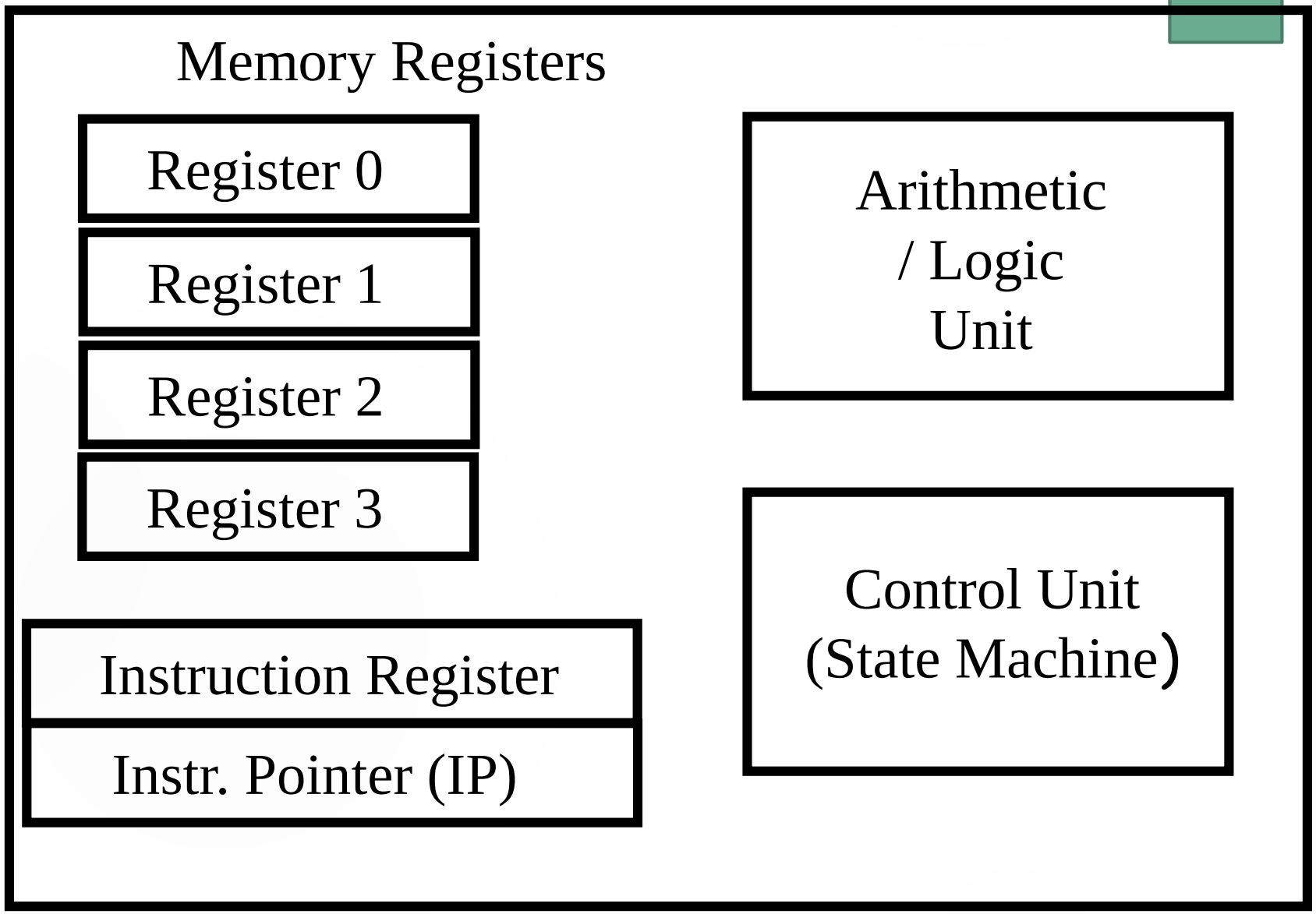
- Arithmetic Logic Unit
  - Status flags
  - Shifter
  - Complementer
  - Arithmetic logic
  - Boolean logic
- Internal CPU bus to pass data back and forth between components of the CPU
- Control unit – managing operation of all CPU components

CPU Internal Structure – Registers

Registers

- CPU must have some working space (temporary storage) to remember things
  - data being operated on
  - pointers to memory (code, stack, data)
  - machine code of current instruction
- Number and function vary between processor designs
- This is one of the major design decisions
- Absolute top level of memory hierarchy

Inside the CPU (cont.)



The Control Unit

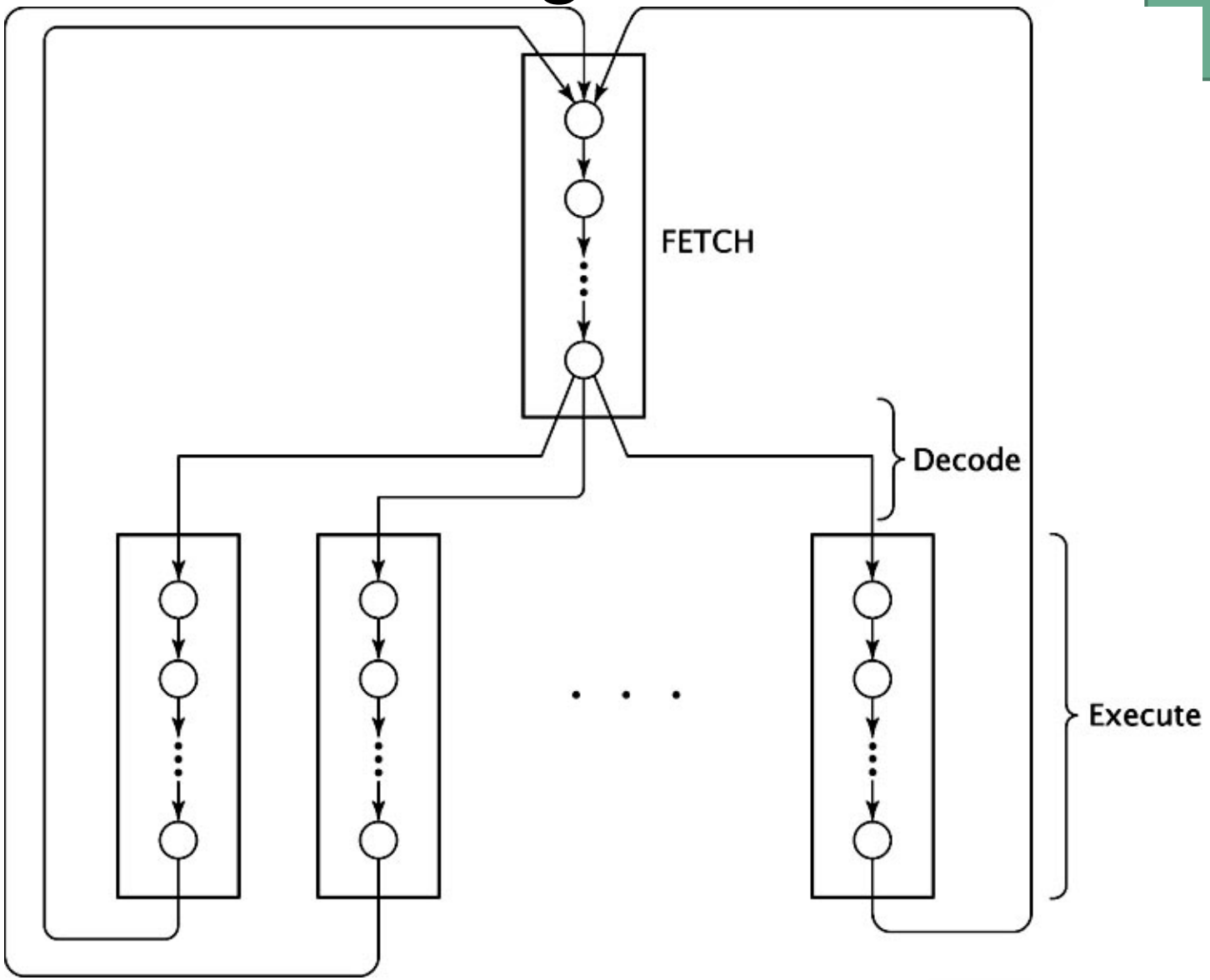
Control Unit State Machine has very simple structure:

- 1) Fetch: Ask the RAM for the instruction whose address is stored in IP.
- 2) Execute: There are only a small number of possible instructions. Depending on which it is, do what is necessary to execute it.
- 3) Repeat: Add 1 to the address stored in IP, and go back to Step 1 !

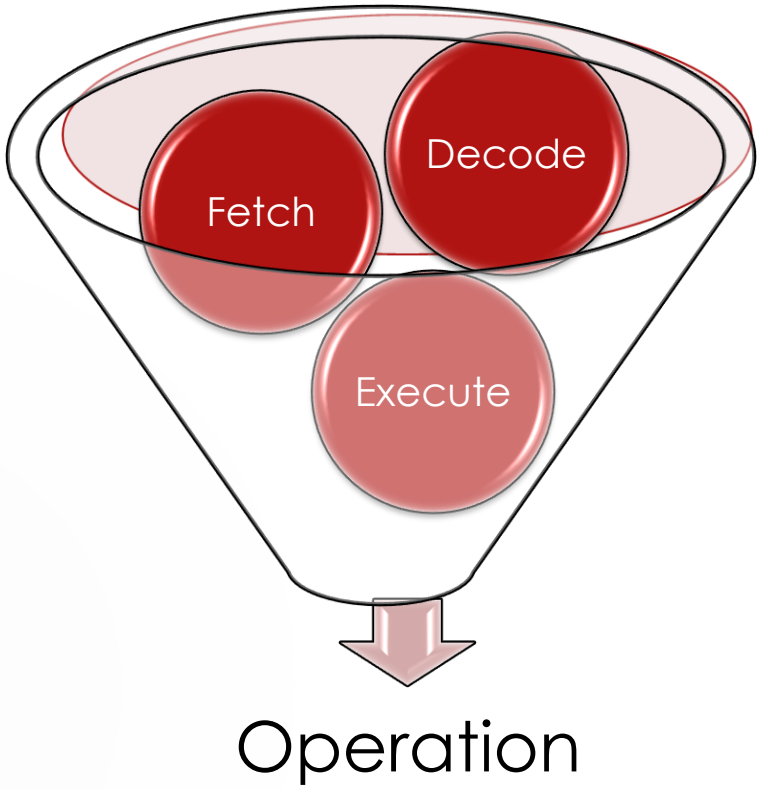
A Simple Program

- Want to add values of variables a and b (assumed to be in memory), and put the result in variable c in memory, I.e. c ← a+b
- Instructions in program
  - Load a into register r1
  - Load b into register r3
  - r2 ← r1 + r3
  - Store r2 in c

CPU State Diagram



Instruction Cycle

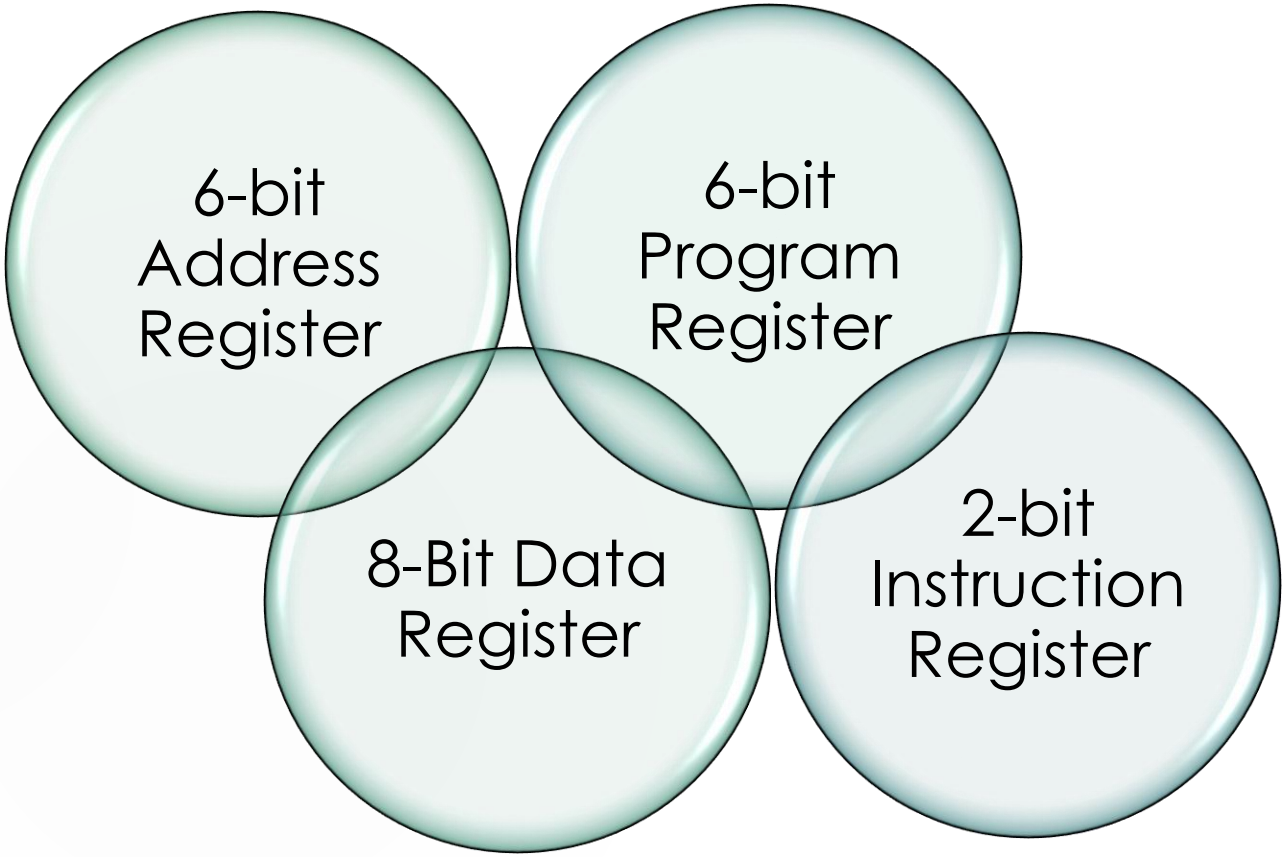


Very Simple CPU

- 64 x 8 memory
  - Address pins A[5..0]
  - Data Pins D[7..0]
- 8-bit Accumulator

Instruction	Instruction Code	Operation
ADD	00AAAAAA	AC←AC + M[AAAAAA]
AND	01AAAAAA	AC←AC ^ M[AAAAAA]
JMP	10AAAAAA	GOTO AAAAAA
INC	11XXXXXX	AC←AC + 1

Non-ISA Registers



Fetch States

FETCH1: AR ← PC

FETCH2: DR ← M, PC ← PC + 1

FETCH3: IR ← DR[7..6], AR ← DR[5..0]

Very Simple CPU Specification

FETCH1: AR ← PC

FETCH2: DR ← M, PC ← PC + 1

FETCH3: IR ← DR[7..6], AR ← DR[5..0]

ADD1: DR ← M

ADD2: AC ← AC + DR

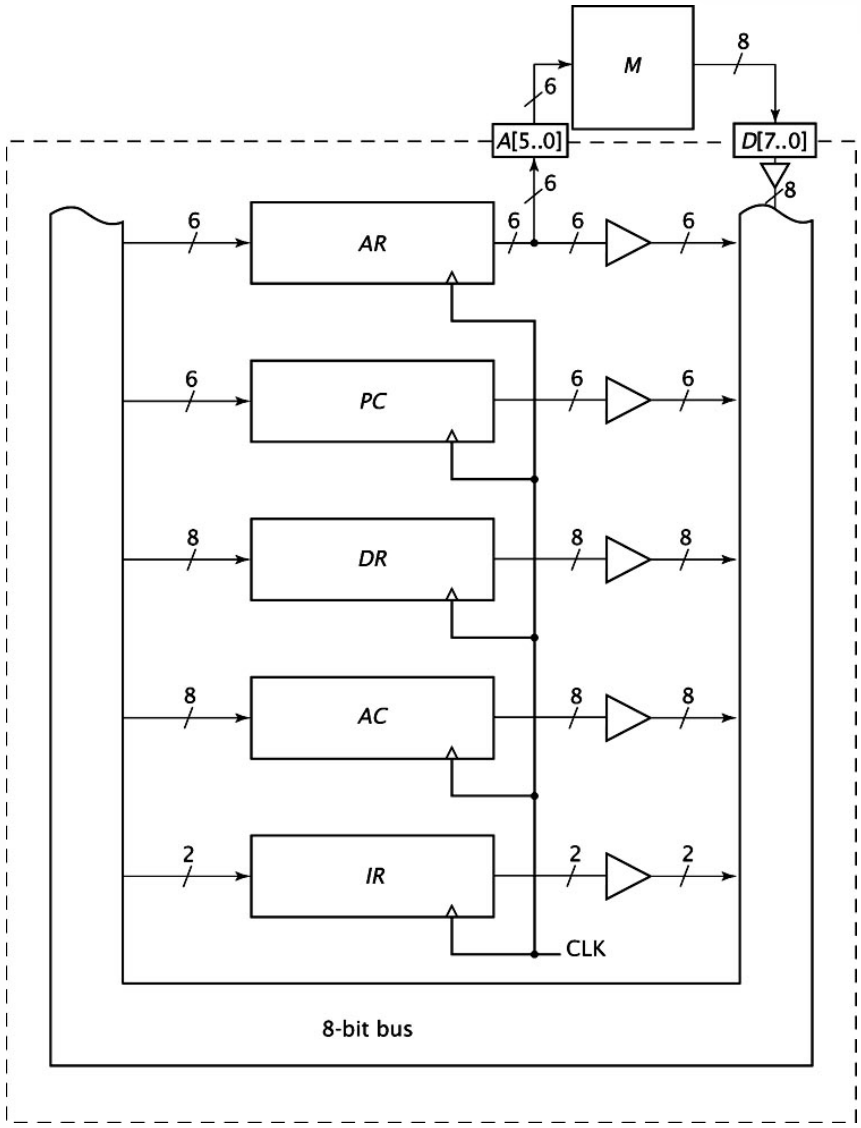
AND1: DR ← M

AND2: AC ← AC ^ DR

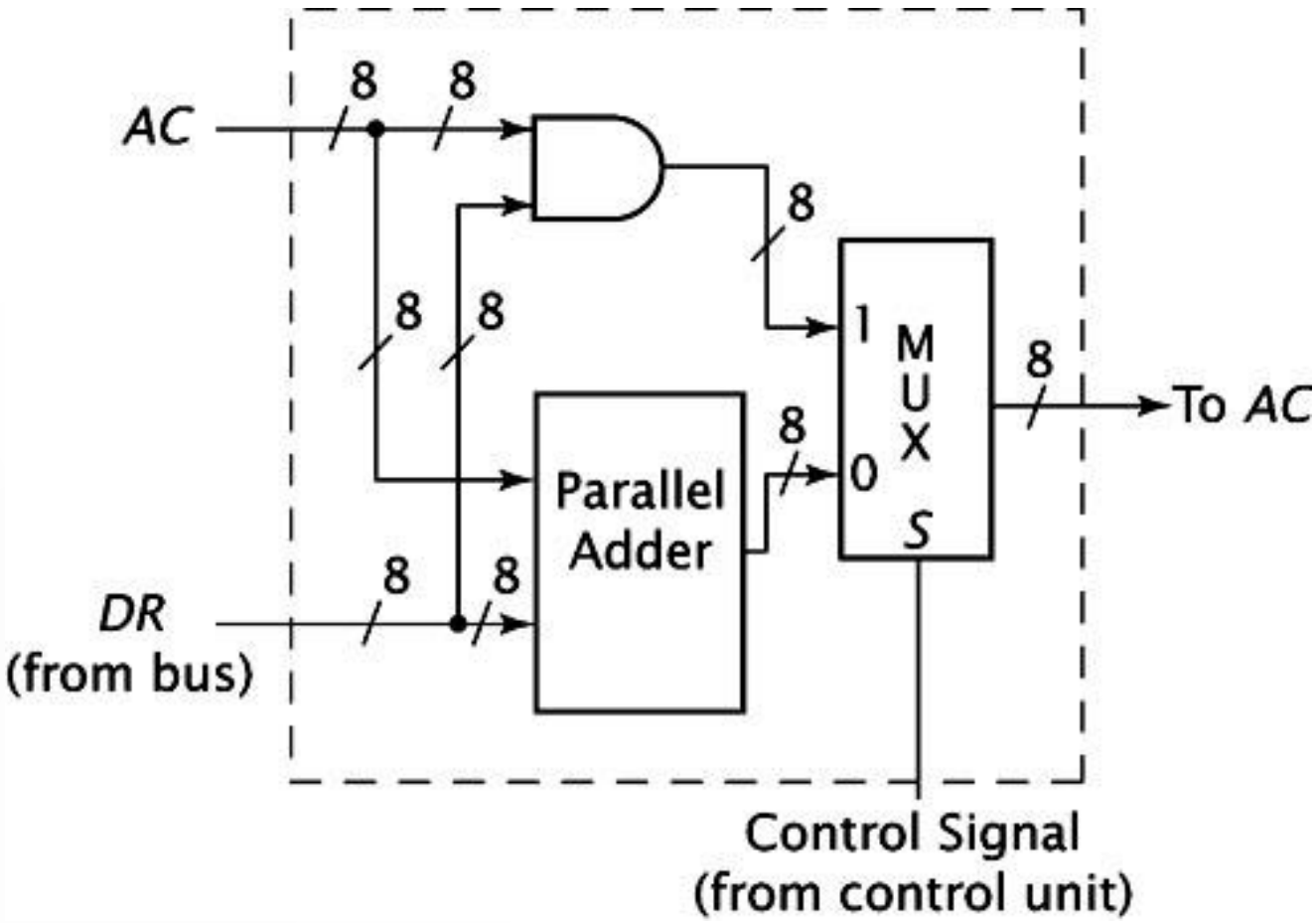
JMP1: PC ← DR[5..0]

INC1: AC ← AC + 1

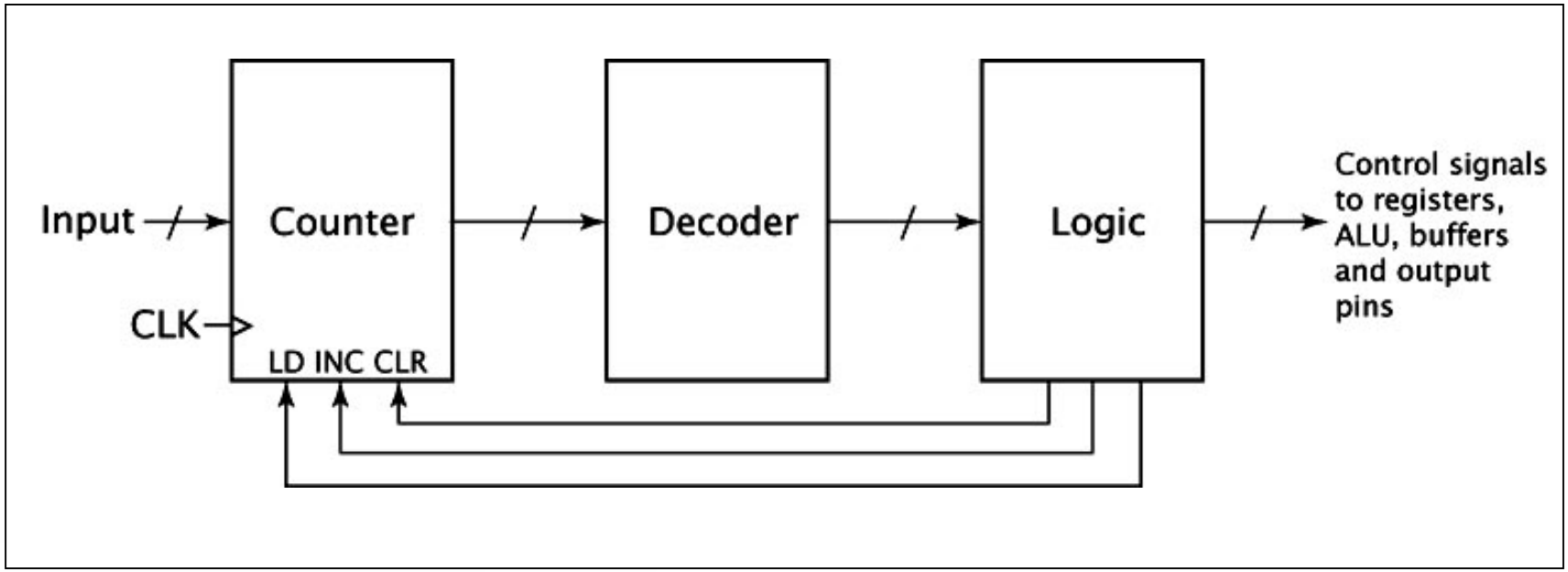
Preliminary Register Section



Very Simple ALU



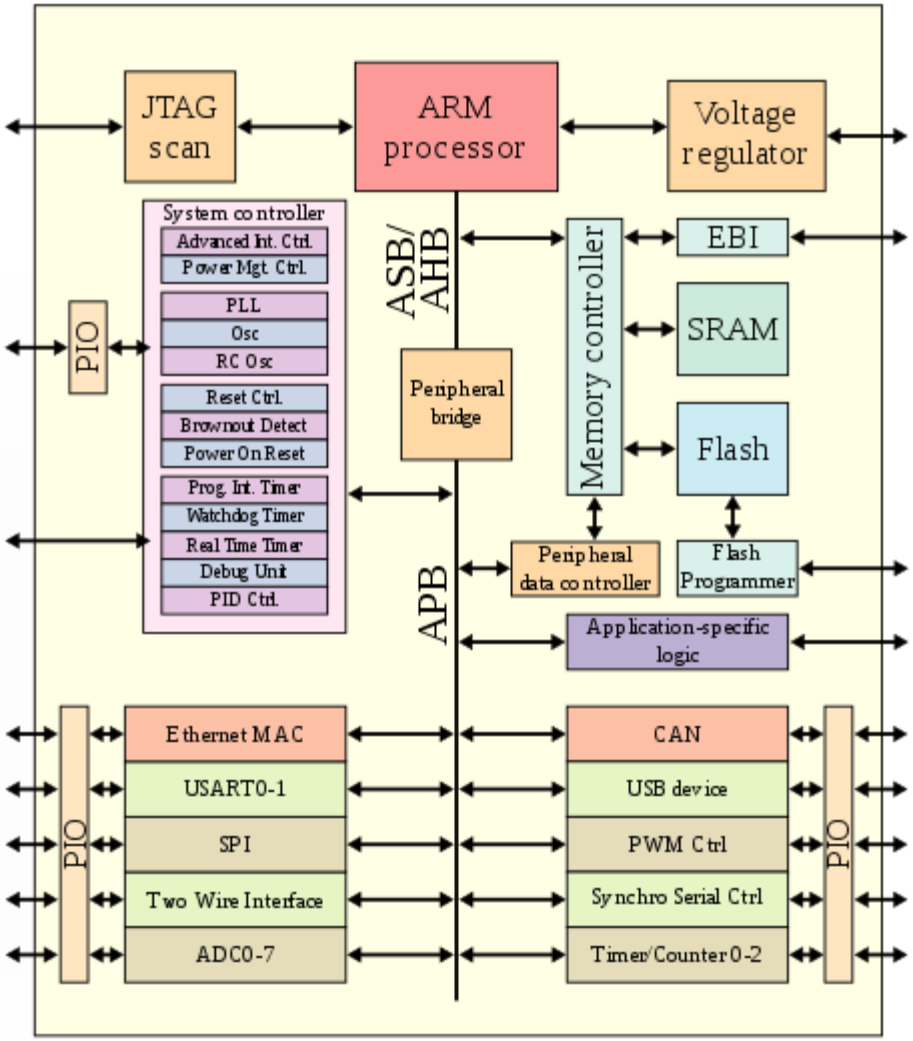
Generic Hardwired Control Unit



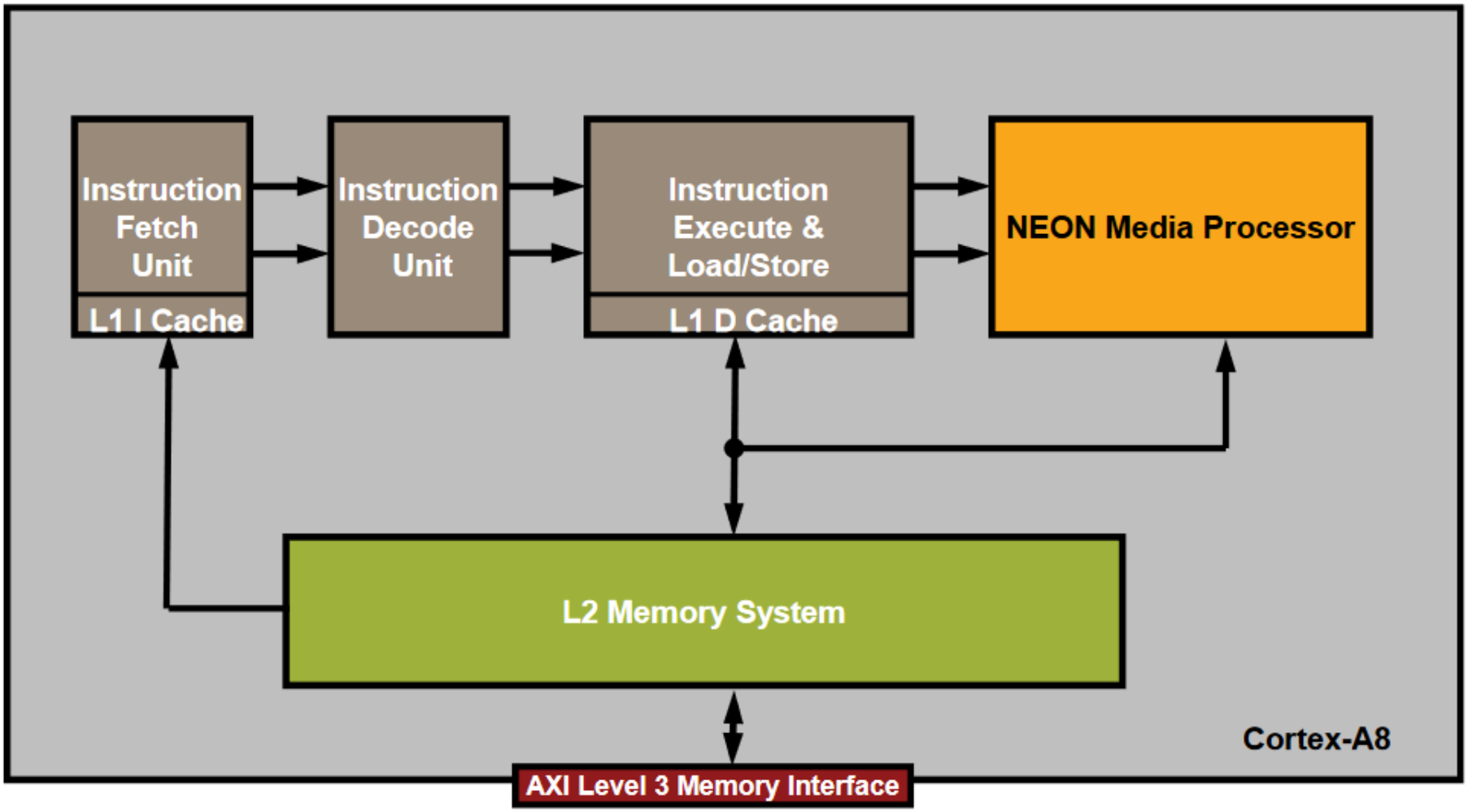
What is ARM Architecture

- This is a family of processor architectures that use RISC
- Has been named Advanced RISC Machine, and even Acorn RISC Machine
- So what is RISC? RISC (Reduced Instruction Set) is an instruction set architecture that allows for fewer cycles per instruction (CPI). The term "reduced" is intended to describe the fact that the amount of work any single instruction accomplishes is reduced. Not that there are fewer instructions. In fact the PowerPC RISC chip has a set of instructions as large as the CISC IBM System/370.
- Processor average for RISC throughput nears 1 instruction per cycle

ARM Architecture



ARM Cortex-A8 Block Diagram



ARM Cortex-A8 Processor Modes

- User - used for executing most application programs
- FIQ - used for handling fast interrupts
- IRQ - used for general-purpose interrupt handling
- Supervisor - a protected mode for the Operating System
- Undefined - entered upon Undefined Instruction exceptions
- Abort - entered after Data or Pre-fetch Aborts
- System - privileged user mode for the Operating System
- Monitor - a secure mode for TrustZone

For more on Cortex-A8 See https://www.arm.com/files/pdf/ARM_Arch_A8.pdf

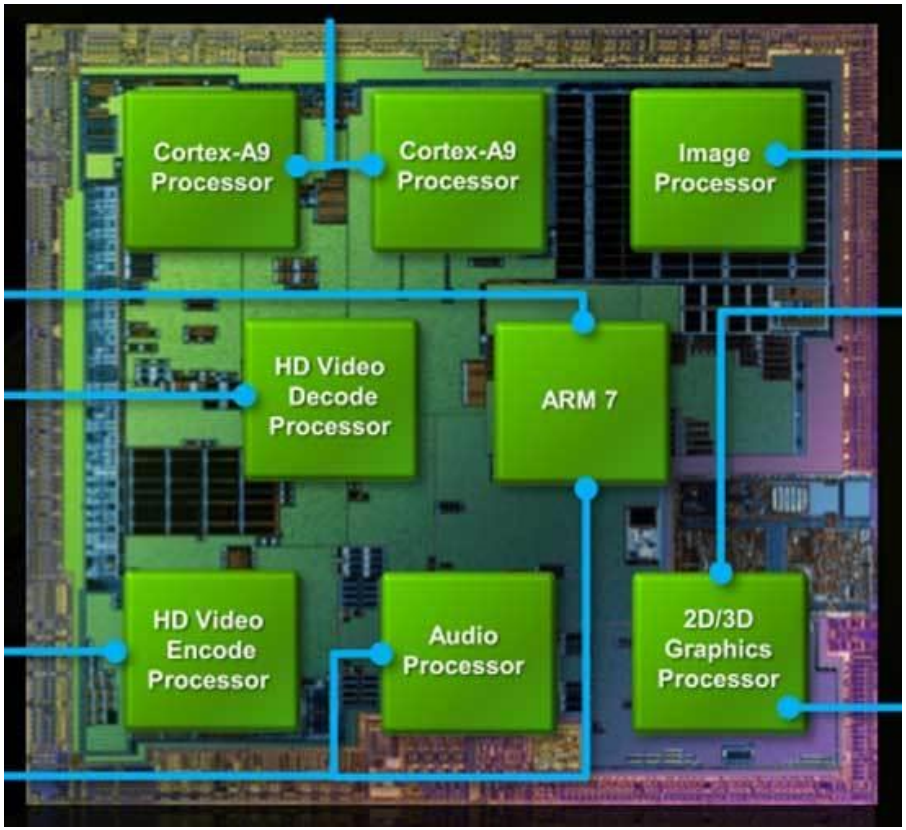
ARMv8-A

- Uses 31 general purpose 64-bit registers
- The instructions, however, are still primarily 32 bit instructions.
- Addresses are 64-bit.
- Fully IEEE 754 compliant (IEEE Standard for Floating-Point Arithmetic)
- Supports double-precision floating point
- ARMv8 introduced hardware acceleration for cryptography.
- Memory translation from 48-bit virtual addresses based on the existing Large Physical Address Extension (LPAE) This is an extension of the ARM 7 architecture that provides an address translation system.

For more see
https://static.docs.arm.com/ddi0487/db/DDI0487D_b_armv8_arm.pdf?_ga=2.19480824.154915068.1558901145-1403176684.1558901145

Graphics Processing Unit (GPU)

- System-on-a-Chip
- Low power consumption
- Accelerate 3D rendering
- Media decoding

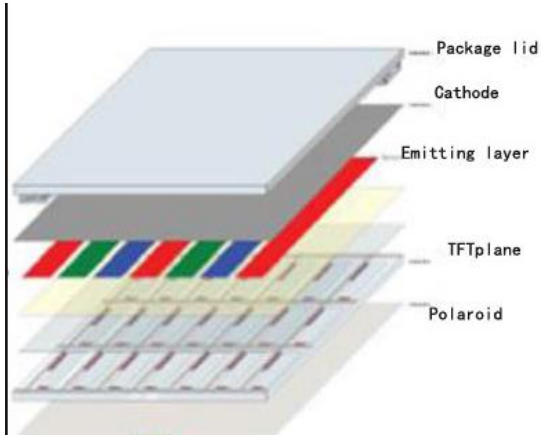


Memory and Storages

- RAM (Random Access Memory)
  - fast
  - Lose contents when power-off
- Internal storage and ROM
  - Store the operating system and critical files
  - Store user and app data
  - Fast access
- Removable storage
  - Supported by Android only
  - Store user and app data
  - Not always available

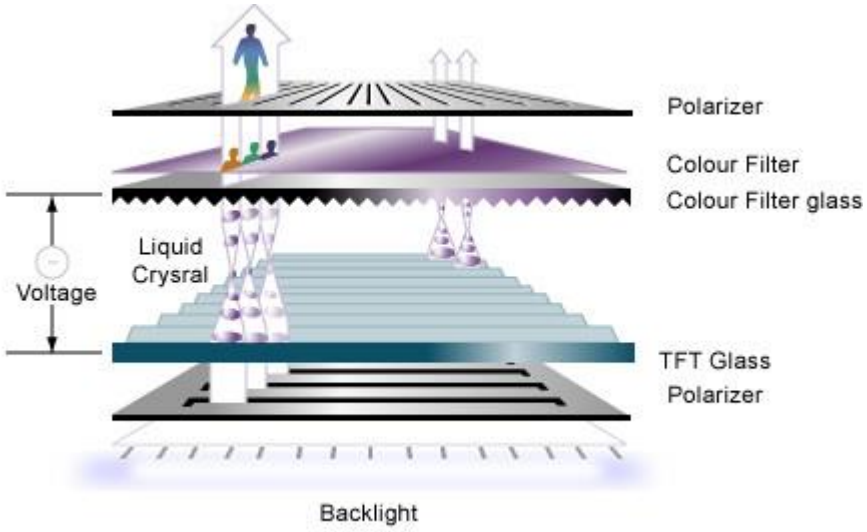
Display Technologies

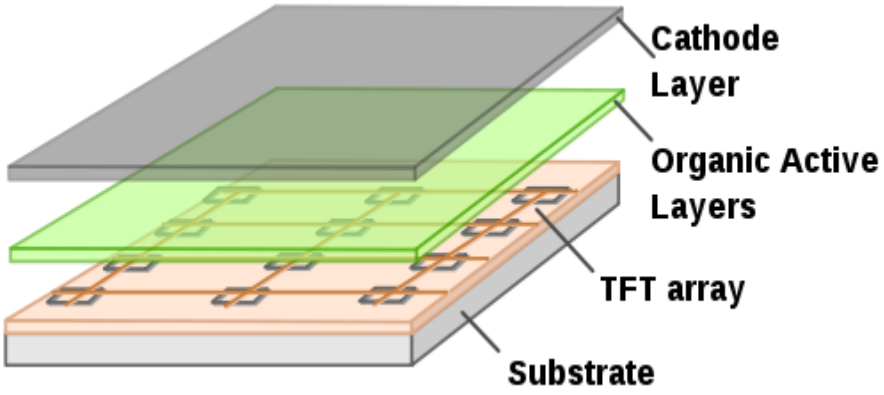
- TWO main types (with subtypes)
  - LCD (Liquid Crystal Display)
  - AMOLED (Active-Matrix Organic Light-Emitting Diode)



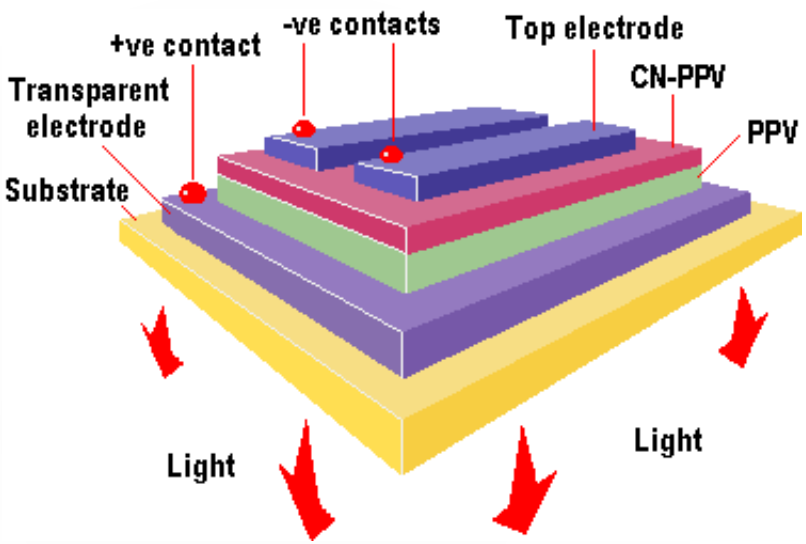
Display Types

- Type one: LCD
  - Cheap to produce
  - Accurate colour reproduction
  - Need backlight
  - Limited viewing angles
- Type two: AMOLED
  - Actively emits colors
  - Good colors and high contrast
  - Good viewing angles
  - Shorter lifespan than LCDs





OLED

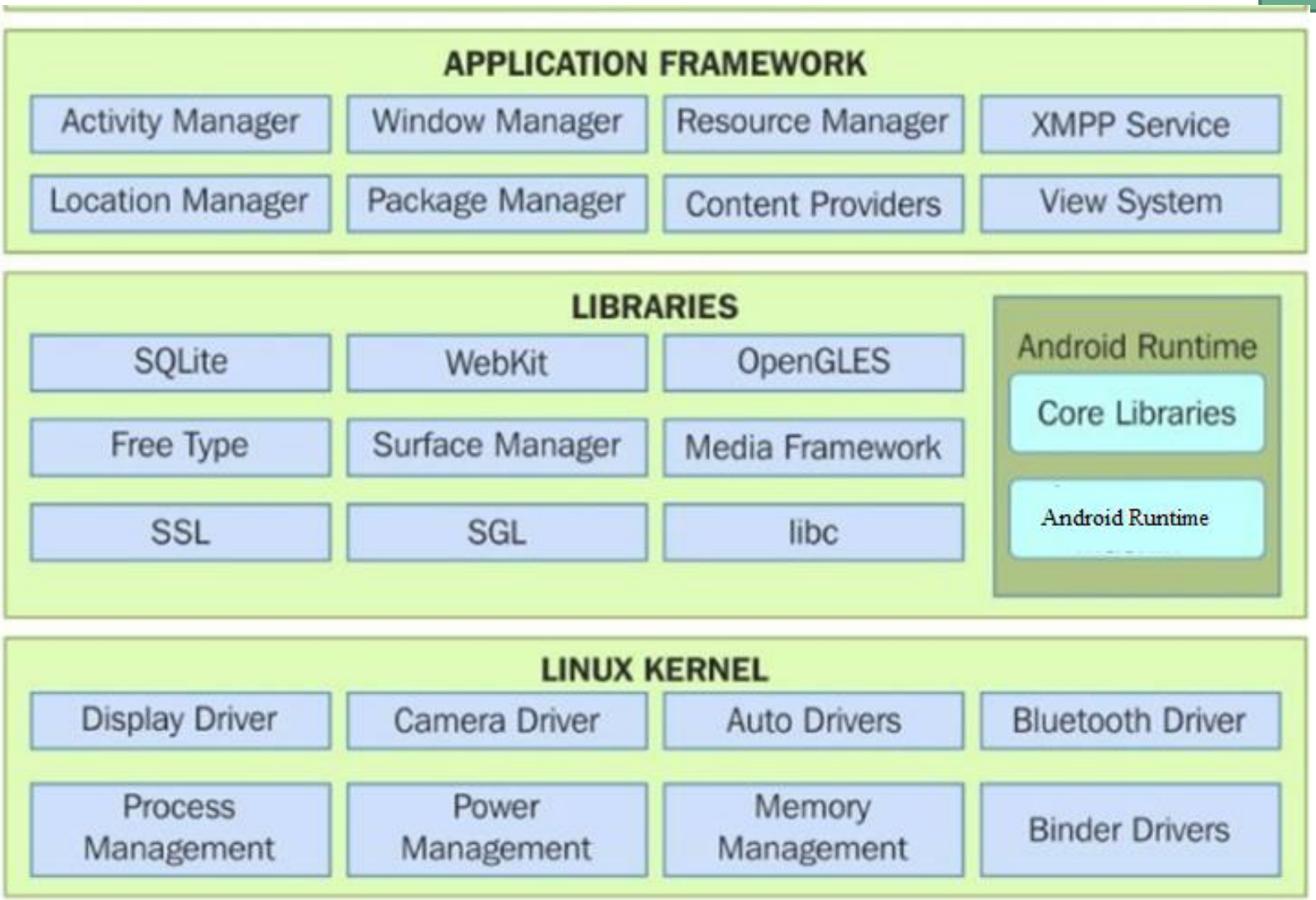


- Already in small sizes
- No inherent size limit
- Conformal displays
- Large viewing angle
- High resolution
- High Speed

Batteries

- Lithium Polymer (Li-Poly) batteries have all of the attributes of a Lithium Ion battery but with ultra slim geometry and simplified packaging. They are the latest technology and found only in few mobile devices.
- Lithium Ion (Li-Ion) batteries are the most popular batteries used in cell phones, as they are light and portable.
- Nickel Metal Hydride (NiMH) batteries are the same as nickel cadmium batteries, but can contain higher energy and can run for between 30 and 40 percent longer.
- Nickel Cadmium (NiCd) batteries are old technology batteries and suffer from memory effect. As a result, the overall capacity and life span of the battery are reduced.

Android Architecture



Android Architecture

Android devices have two primary types of memory:

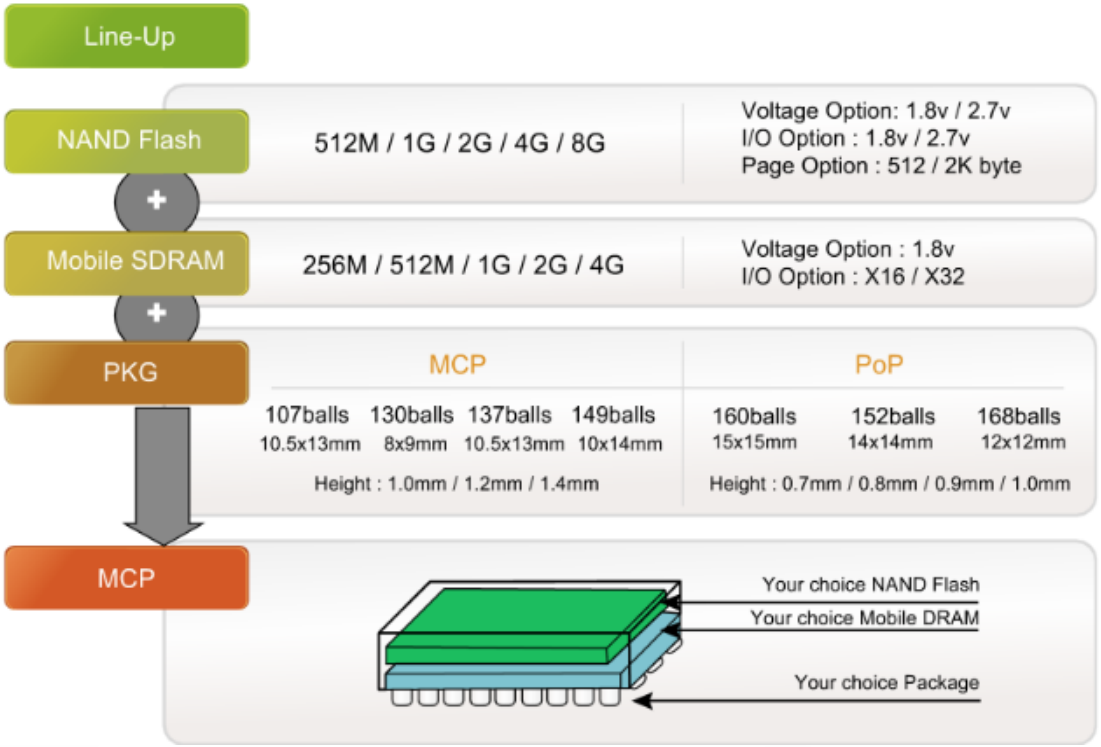
- Volatile: random-access memory (RAM)
- Non-volatile: NAND flash memory

NAND flash

Critical for forensic investigation

MCP

RAM and NAND in a single component named multichip package



Linux Kernel

- As of 2018, Android targets versions 4.4, 4.9 or 4.14 of the Linux kernel.
- Does not include the full set of standard Linux utilities
- No native windowing system
- Used as a hardware abstraction layer
- Performs important power management activities
- Open Source: provide libraries to modify hardware drivers

Linux Kernel

Android Version	Linux Kernel
1.0 to 3.0	used 2.6.xx
4.0 to 6.0	used 3.xx
7.0 to 9.0	used 4.x

Android OS

Memory management

- Android manages the apps stored in memory automatically: when memory is low, the system will begin killing apps and processes that have been inactive for a while, in reverse order since they were last used (i.e. oldest first).
- This process is designed to be invisible to the user, such that users do not need to manage memory or the killing of apps themselves.

Android App Priority and Processes

- Android apps do not have control over their own life cycles
- Aggressively manages resources to ensure device responsiveness and kills process/apps when needed

- Active Process – critical priority
- Visible Process – high priority
- Started Service Process
- Background Process – low priority
- Empty process

Linux Kernel and Storage Management

- The flash storage on Android devices is split into several partitions, such as /system for the operating system itself, and /data for user data and application installations.
- In contrast to desktop Linux, Android device owners are not given root access to the operating system and sensitive partitions such as /system are read-only. However, root access can be obtained by exploiting security flaws in Android.

Boot process

- Before the device is powered on, the device CPU will be in a state where no initializations have been done. Once the Android device is powered on, execution starts with the boot ROM code. This boot ROM code is specific to the CPU the device is using. As demonstrated in the following screenshot, this phase includes two steps, A and B:
  - Step A: When the boot ROM code is executed, it initializes the device hardware and tries to detect the boot media. Hence, the boot ROM code scans until it finds the boot media.
  - Step B: Once the boot sequence is established, the initial boot loader is copied to the internal RAM. After this, the execution shifts to the code loaded into RAM.
  - -Learning Android Forensics: Analyze Android devices with the latest forensic tools and techniques, 2nd Edition

Boot Loader

- The bootloader is a small program that is executed before the operating system starts to function. Bootloaders are present in desktop computers, laptops, and mobile devices as well. In the Android boot loader, there are two stages—Initial Program Load (IPL) and Second Program Load (SPL). As shown in the following screenshot, this involves the three steps explained here:
  - Step A: IPL deals with detecting and setting up the external RAM.
  - Step B: Once the external RAM is available, SPL is copied into the RAM and execution is transferred to it. SPL is responsible for loading the Android operating system.
  - Step C: SPL tries to look for the Linux kernel. It will load this from boot media and will copy it to the RAM. Once the boot loader is done with this process, it transfers the execution to the kernel:

-Learning Android Forensics: Analyze Android devices with the latest forensic tools and techniques, 2nd Edition

Linux Kernel

- The Linux kernel is the heart of the Android operating system and is responsible for process management, memory management, and enforcing security on the device. After the kernel is loaded, it mounts the root filesystem (rootfs) and provides access to system and user data:
  - Step A: When the memory management units and caches have been initialized, the system can use virtual memory and launch user space processes.
  - Step B: The kernel will look in the rootfs for the init process and launch it as the initial user space process:
- -Learning Android Forensics: Analyze Android devices with the latest forensic tools and techniques, 2nd Edition

Process descriptions

- The process rootfs is the place to start and stop searching the doubly linked list of mount points. It is a special instance of ramfs which is a simple file system used to get the system booted.
- The basic initramfs is the root filesystem image used for booting the kernel provided as a compressed archive.
  - If an uncompressed cpio archive exists at the start of the initramfs, extract and load the microcode from it to CPU.
  - If an uncompressed cpio archive exists at the start of the initramfs, skip that and set the rest of file as the basic initramfs. Otherwise, treat the whole initramfs as the basic initramfs.
  - unpack the basic initramfs by treating it as compressed (currently gzipped) cpio archive into a RAM-based disk.
  - mount and use the RAM-based disk as the initial root filesystem.

What is a Kernel

- Core or nucleus of an operating system
- Interacts with the hardware
- First program to get loaded when the system starts and runs till the session gets terminated
- Different from BIOS which is hardware dependent.
- Kernel is software dependent

Kernel types

- Monolithic
  - All OS related code are stuffed in a single module
  - Available as a single file
  - Advantage : Faster functioning
  - Note: Linux has a monolithic kernel
- Micro
  - OS components are isolated and run in their own address space
  - Device drivers, programs and system services run outside kernel memory space
  - Supports modularity
  - Lesser in size

What is a Shell

- Program that interacts with kernel
- Bridge between kernel and the user
- Command interpreter
- User can type command and the command is conveyed to the kernel and it will be executed

What is a Link

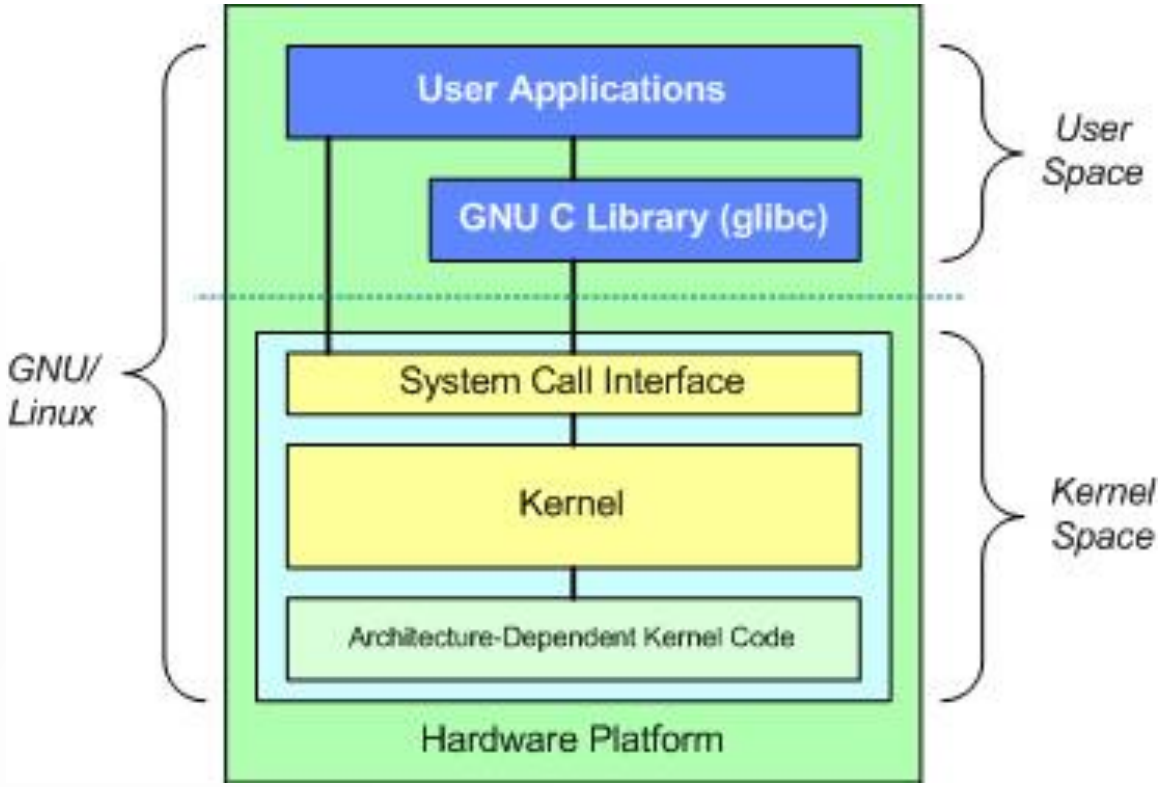
- Soft or symbolic links are just like hard links. It allows to associate multiple filenames with a single file. However, symbolic links allows:
  - To create links between directories
  - Can cross file system boundaries
- These links behave differently when the source of the link is moved or removed.
  - Symbolic links are not updated.
  - Hard links always refer to the source, even if moved or removed

Concept Links - continued

- Symbolic links (also called symlinks or softlinks) most resemble Windows shortcuts. They contain a pathname to a target file. Hard links are a bit different. They are listings that contain information about the file. Linux files don't actually live in directories. They are assigned an *inode* number, which Linux uses to locate files. So a file can have multiple hardlinks, appearing in multiple directories, but isn't deleted until there are no remaining hardlinks to it. Here are some other differences between hardlinks and symlinks:

1. You cannot create a hardlink for a directory.
2. If you remove the original file of a hardlink, the link will still show you the content of the file.
3. A symlink *can* link to a directory.
4. A symlink, like a Windows shortcut, becomes useless when you remove the original file.

Linux Kernel



For more information http://www.ibm.com/developerworks/linux/library/l-linux-kernel/?S_TACT=105AGX59&S_CMP=GR&ca=dgr-btw01LKernalAnatomy

Memory Management

- One very important function of the kernel is memory management. Since the kernel has full access to the system's memory, one of the kernels functions is to allow processes to safely access this memory as those processes require it. This is often accomplished with techniques such as virtual addressing, usually achieved by paging and/or segmentation. Virtual addressing allows the kernel to make a given physical address appear to be another address, the virtual address. Virtual address spaces may be different for different processes; the memory that one process accesses at a particular (virtual) address may be different memory from what another process accesses at the same address. This allows every program to behave as if it is the only one (apart from the kernel) running and thus prevents applications from crashing each other

Interupts

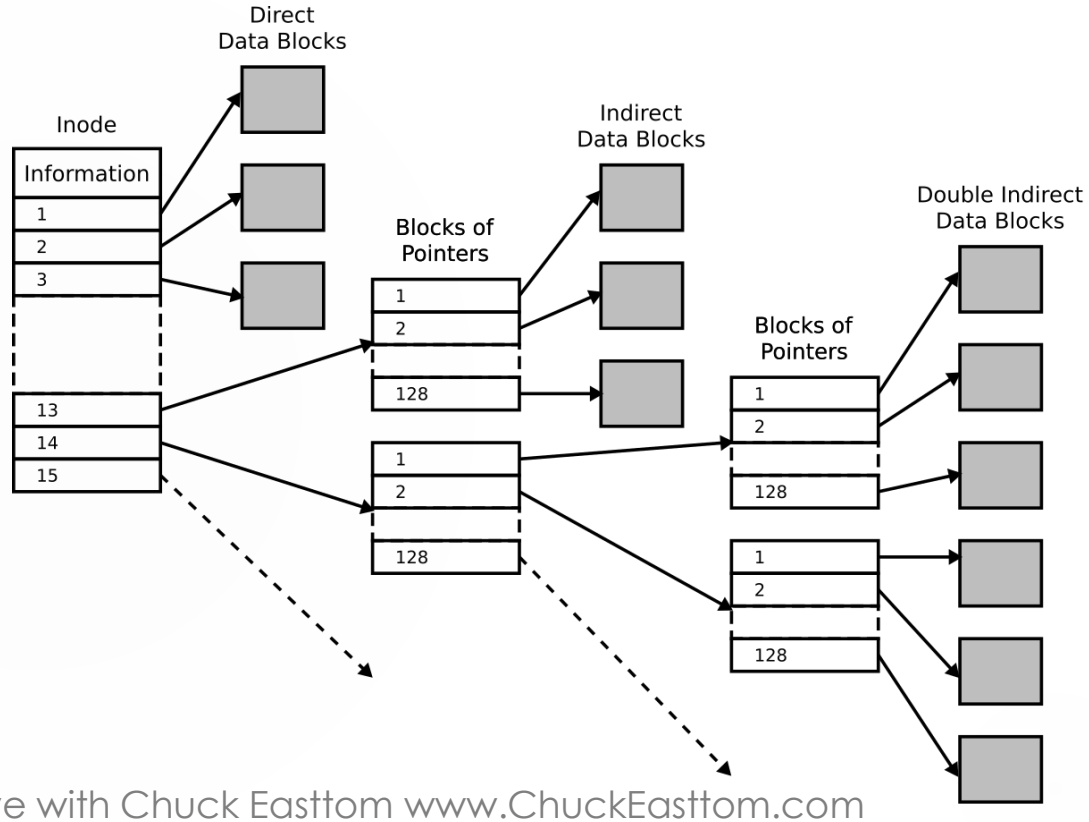
- Interrupts, put simply, are pathways by which the various hardware devices or programs can communicate with the operating system. A more technical definition would be that an interrupt is an asynchronous signal indicating the need for attention or an event in software indicating the need for a change in execution.

Device Drivers

- Device drivers are simply programs that provide an interface between the operating system and some hardware, such as a printer. Normally operating systems ship with a number of device drivers. It is also common practice for hardware vendors to include drivers as part of their installation programs

inodes

- An inode is a data structure in the file system that stores all the information about a file except its name and its actual data.
- Whenever you refer to a file by name, perhaps to open a text document, a process is initiated. That process involves the operating system using the file name to look up the corresponding inode, which then enables the system to obtain the information it needs about the file to perform further operations



inode - continued

- An inode can refer to a file or a folder/directory. In either case, the inode is really a link to the file. This is important because there are basically two types of links. The first type is the *hard link*. A hard link is an inode that links directly to a specific file. The operating system keeps a count of references to this link. Whecdn the reference count reaches zero, the file is deleted. In other words, you can have any number of names referencing a file, but if that number of references reaches zero (i.e., there is *no* name that references that file), then the file is deleted.

The Android File system

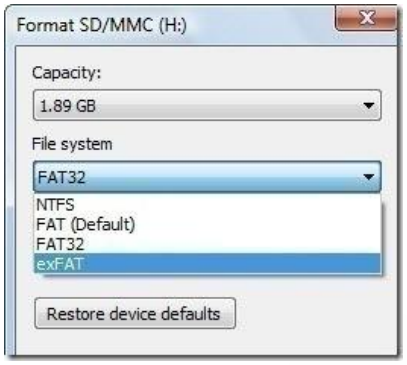
Can use any of these

- **exFAT** - The extended File Allocation Table is a Microsoft proprietary file system for flash memory. Due to the licensing requirements, it is not part of the standard Linux kernel. However, some manufactures provide Android support for the file system.
- **F2FS** - Samsung introduced The Flash-Friendly File System as an open source Linux file system in 2012.
- **JFFS2** - The Journal Flash File System version 2 is the default flash file system for the AOSP (Android Open Source Project) kernels, since Ice Cream Sandwich. JFFS2 is a replacement to the original JFFS.
- **YAFFS2** - Yet Another Flash File System version 2 was the default AOSP flash file system for kernel version 2.6.32. YAFFS2 is not supported in the newer kernel versions, and does not appear in the source tree for the latest kernel versions from kernel.org. However, individual mobile device vendors may continue to support YAFFS2.

.

exFAT

- The extended File Allocation was introduced by Microsoft in 2006 and optimized for flash memory
- File size limit of 16 exabytes
- Can scale to very large disks
- exFat employs a filename hash-based lookup phase to speed certain cases, which is described in US Patent 8321439



F2FS Flash-Friendly File System

- Developed by Samsung specifically for the Linux kernel
- Divides the volume into 2 megabyte segments
- Zones consists of a set of segments.
- Entire volume is divided into six areas

F2FS Flash-Friendly File System

- **Superblock (SB)**
  - The SB is located at the beginning of the partition. There are two copies to avoid file-system corruption. It contains basic partition information and some default F2FS parameters.
- **Checkpoint (CP)**
  - The CP contains file system information, bitmaps for valid NAT/SIT sets, orphan inode lists, and summary entries of current active segments.
- **Segment Information Table (SIT)**
  - The SIT contains the valid block count and validity bitmap of all the Main Area blocks.
- **Node Address Table (NAT)**
  - The NAT is an address table for the Main Area node blocks.
- **Segment Summary Area (SSA)**
  - The SSA contains entries which contain the owner information of the Main Area data and node blocks.
- **Main Area**
  - The main area contains file and directory data and their indices

F2FS Flash-Friendly File System

- The key data structure is the "node". Similar to traditional file structures, F2FS has three types of nodes: inode, direct node, indirect node.
- 2FS assigns 4 KB to an inode block which contains 923 data block indices, two direct node pointers, two indirect node pointers, and one double indirect node pointer as described below. A direct node block contains 1018 data block indices, and an indirect node block contains 1018 node block indices.

hash	hash value of the file name
ino	inode number
len	the length of file name
type	file type such as directory, symlink, etc

F2FS Flash-Friendly File System

A directory entry structure

hash	hash value of the file name
ino	inode number
len	the length of file name
type	file type such as directory, symlink, etc

Block Allocation Policy

Hot node	Contains direct node blocks of directories.
Warm node	Contains direct node blocks except hot node blocks.
Cold node	Contains indirect node blocks.
Hot data	Contains dentry blocks.
Warm data	Contains data blocks except hot and cold data blocks.
Cold data	Contains multimedia data or migrated data blocks.

JFFS2

About JFFS2:

- Log Structured File System to enable write leveling
- On-the-fly compression and decompression
- Uses Cleanmarkers to indicate whether a block was successfully erased

0x08	0x09	0x0a	0x0b	0x0c	0x0d	0x0e	0x0f
0x85	0x19	0x03	0x20	0x08	0x00	0x00	0x00

YAFFS

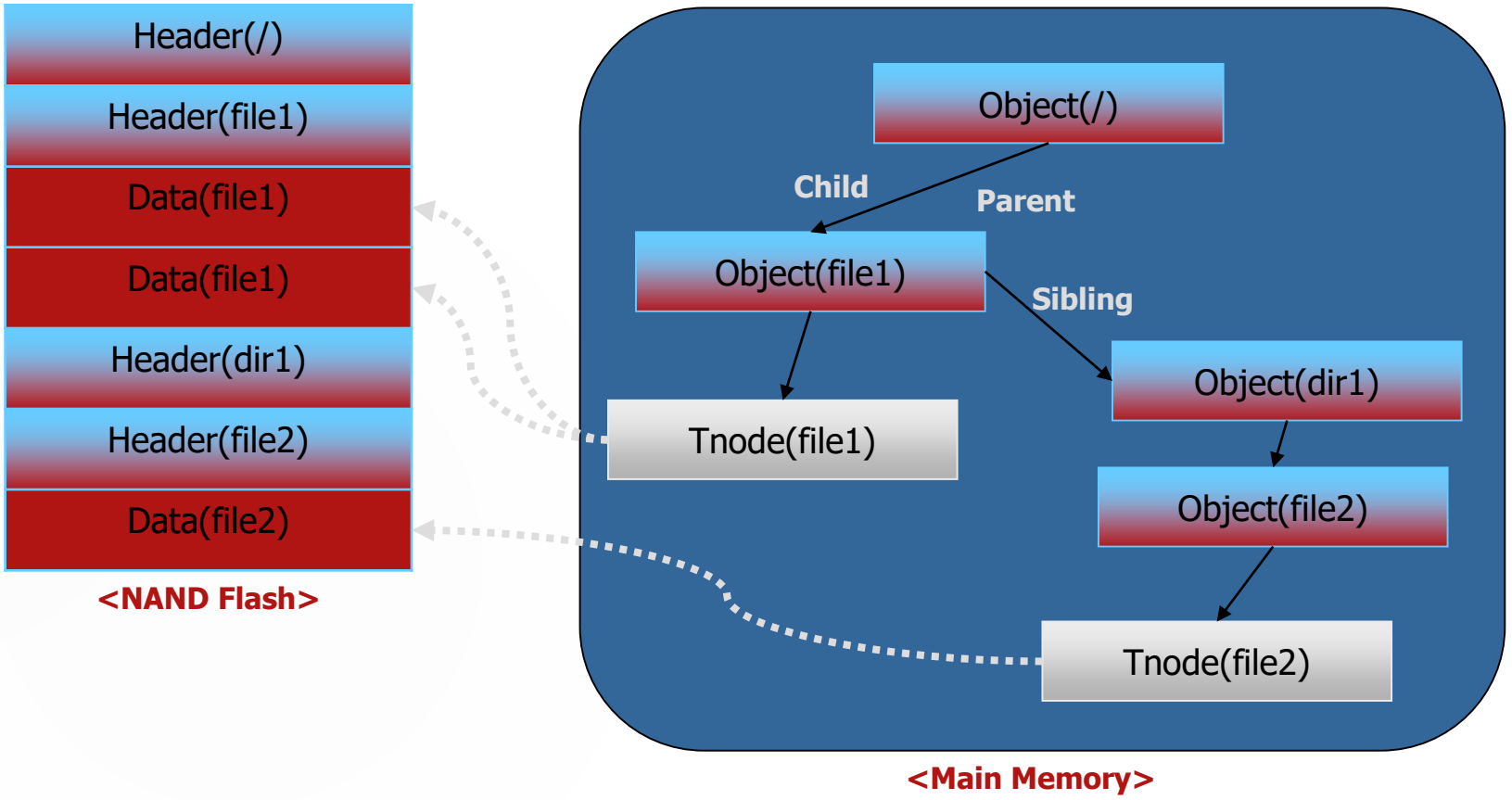
- Designed by Charles Manning of New Zealand, first released in 2002. Uses 512 byte page size. Designed for NAND flash chips.

- YAFFS2 was an improvement. 'Chunks are identified by ObjectID's and ChunkNumbers.

YAFFS Terminology

- Flash-defined
  - Page - 2k flash page (512 byte YAFFS1)
  - Block - Erasable set of pages (typically 64 on 2K NAND)
- YAFFS-defined
  - Chunk - YAFFS tracking unit.
  - Usually == page.

YAFFS Layout | RAM



YAFFS Log-structured FS(1)

- Supposed the flash with 4 pages per block
- First, Create the file

Block	Chunk	ObjID	ChunkID	Delete?	Comment
0	0	100	0	Live	Object header for this files

Then, Write a few chunks

Block	Chunk	ObjID	ChunkID	Delete?	Comment
0	0	100	0	Live	Object header for this files
0	1	100	1	Live	First chunk of data
0	2	100	2	Live	Second chunk of data
0	3	100	3	Live	Third chunk of data

YAFFS Log-structured FS(2)

- Close the file >> Write a new object header for the file

Block	Chunk	ObjID	ChunkID	Delete?	Comment
0	0	100	0	Del	Obsoleted object header
0	1	100	1	Live	First chunk of data
0	2	100	2	Live	Second chunk of data
0	3	100	3	Live	Third chunk of data
1	0	100	0	Live	New object header

YAFFS Log-structured FS(3)

- Open the file for read/write and overwrite part of the first chunk in the file and close the file. The replaced data and object header chunks become deleted.

Block	Chunk	ObjID	ChunkID	Delete?	Comment
0	0	100	0	Del	Obsoleted object header
0	1	100	1	Del	Obsoleted First chunk of data
0	2	100	2	Live	Second chunk of data
0	3	100	3	Live	Third chunk of data
1	0	100	0	Del	Obsoleted object header
1	1	100	1	Live	New first chunk of data
1	2	100	0	Live	New object header

YAFFS Log-structured FS(4)

- Now resize the file to zero by opening the file with O_TRUNC and closing the file. This writes a new object header with length 0 and marks the data chunks deleted.

Block	Chunk	ObjID	ChunkID	Delete?	Comment
0	0	100	0	Del	Obsoleted object header
0	1	100	1	Del	Obsoleted First chunk of data
0	2	100	2	Del	Obsoleted 2nd chunk of data
0	3	100	3	Del	Obsoleted 3rd chunk of data
1	0	100	0	Del	Obsoleted object header
1	1	100	1	Del	Obsoleted first chunk of data
1	2	100	0	Del	Obsoleted object header
1	3	100	0	Live	New object header

YAFFS Log-structured FS(5)

- Block#1 is dirty block Rename the file, then we need new object header

Block	Chunk	ObjID	ChunkID	Delete?	Comment
0	0	100	0	Del	Obsoleted object header
0	1	100	1	Del	Obsoleted First chunk of data
0	2	100	2	Del	Obsoleted 2nd chunk of data
0	3	100	3	Del	Obsoleted 3rd chunk of data
1	0	100	0	Del	Obsoleted object header
1	1	100	1	Del	Obsoleted first chunk of data
1	2	100	0	Del	Obsoleted object header
1	3	100	0	Del	Obsoleted object header
2	0	100	0	Live	New object header

YAFFS2

- Designed for new hardware:
  - >=1k page size
  - No re-writing
  - Can support Toshiba/Sandisk MLC parts
- Main difference is ‘discarded’ status tracking
- ECC done by driver (MTD in Linux case)
- Extended Tags (Extra metadata to improve performance)
- RAM footprint 25-50% less
- faster (write 1-3x, read 1-2x, delete 4-34x, GC 2-7x)

OOB (Out of Band) Data

- YAFFS1:
  - Derived from Smartmedia, (e.g byte 5 is bad block marker)
  - 16 bytes: 7 tags, 2 status, 6 ECC
  - YAFFS/Smartmedia or JFFS2 format ECC
- YAFFS2:
  - 64 bytes available in 2k page
  - MTD-determined layout
  - MTD or hardware does ECC
  - Tags normally 28 bytes (16 data, 12ecc)

YAFFS Summary

- Very simple
- NAND-friendly
- Relatively frugal with resources (especially RAM)
- Boot quickly
- Journaling (robustness)

Android Permissions

"Android implements a permission model for individual apps. Applications must declare which permissions (in the manifest file) they require. In older versions of Android, the user was presented with a full list of permissions requested by the application prior to installation.

Newer versions of Android prompt the user the first time each permission is required while the app is in use. This model allows a user to use an app without granting all permissions requested by the application, though functionality may be decreased."

-Learning Android Forensics: Analyze Android devices with the latest forensic tools and techniques, 2nd Edition

Android Permissions

Level	Description
Normal	Low Risk
Dangerous	Approval needed
Signature	must be signed by the same certificate as the one that declared/created the permission.
Signature/System	only for applications in the system image

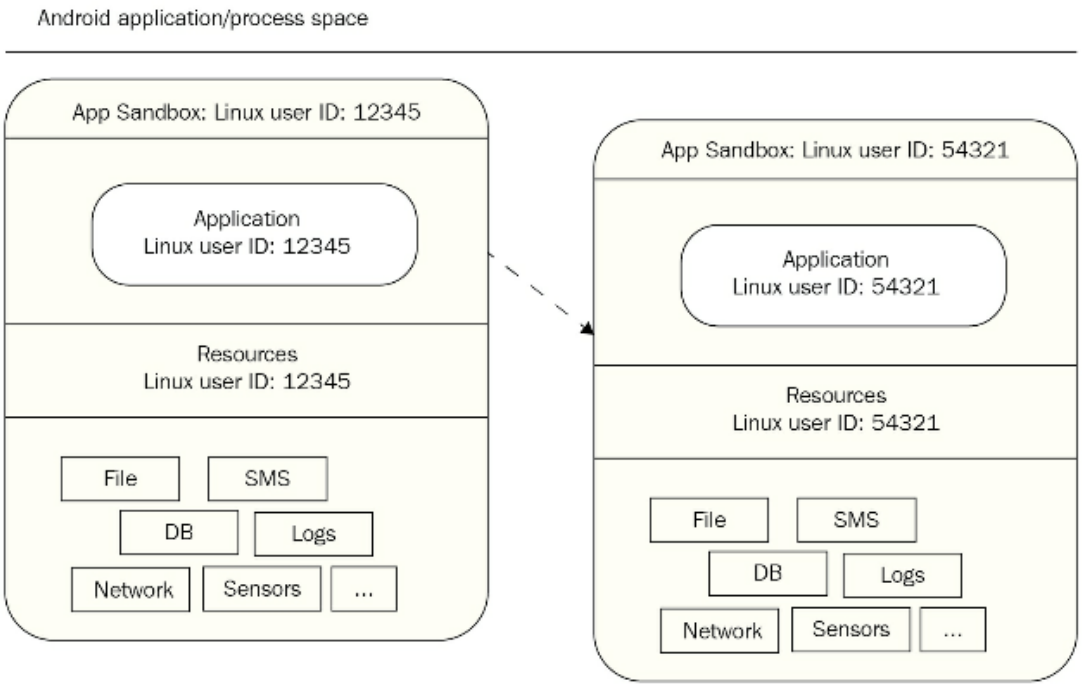
-Learning Android Forensics: Analyze Android devices with the latest forensic tools and techniques, 2nd Edition

Android App Sandboxing

"In order to isolate applications from each other, Android takes advantage of the Linux user-based protection model. In Linux systems, each user is assigned a unique User ID (UID) and users are segregated so that one user does not have access to the data of another. All resources under a particular user are run with the same privileges. Similarly, each Android application is assigned a UID and is run as a separate process. What this means is that, even if an installed application tries to do something malicious, it can do it only within its context and with the permissions it has."

-Learning Android Forensics: Analyze Android devices with the latest forensic tools and techniques, 2nd Edition

Android App Sandboxing



- Learning Android Forensics: Analyze Android devices with the latest forensic tools and techniques, 2nd Edition

Android and SELinux

“Starting with Android 4.3, Security-Enhanced Linux (SELinux) is supported by the Android security model. Android security is based on discretionary access control, which means applications can ask for permissions, and users can grant or deny those permissions. Hence, malware can create havoc on the phones by gaining permissions. Android uses SELinux to enforce mandatory access control that ensures applications work in isolated environments; this includes applications running as root or superuser. Hence, even if a user installs a malicious app, the malware cannot easily access the OS and corrupt the device. SELinux is used to enforce Mandatory Access Control (MAC) over all of the processes, including the ones running with root privileges.

-Learning Android Forensics: Analyze Android devices with the latest forensic tools and techniques, 2nd Edition

Android APP Signing

All apps have to be signed, but Android is not picky about who signs them.

Libraries

- OpenGL ES android.opengl
  - The OpenGL ES is a 3D graphics library.
- SQLite android.database.sqlite
- Contains the SQLite database management classes
- Media Framework
- The media framework contains all of the codecs that are required for multimedia experience.
- FreeType: used to render the fonts
- SSL: used for internet security
- WebKit: open source browser engine

Android Versions

Android was first released in 2003 and is the creation of Rich Miner, Andy Rubin, and Nick Sears. Google acquired Android in 2005, but still keeps the code open source. The versions of Android have been named after deserts/sweets:

- version 1.5 Cupcake
- version 1.6 Donut.
- version 2.0-2.1 Éclair
- version 2.2 Froyo
- version 2.3 Gingerbread
- version 3.1 - 3.2 Honeycomb
- version 4.0 Ice Cream Sandwich
- version 4.1 - 4.2 Jelly Bean
- version 4.3 Kitkat
- Version 5.0 Lollipop released November 3 2014
- Version 6.0 Marshmallow released October 2015
- Version 7.0 Nougat was released August 2016
- Version 8.0 Oreo released August 2017
- Version 9.0 Pie August 6, 2018
- Android Q – Beta 2019 NOW just called by number 10 Scheduled for Release September 3

- Google explained that "Since these devices make our lives so sweet, each Android version is named after a dessert"

Android

Android

- Important Security Features:
  - **Version 9 (Pie):**
  - Lockdown mode disables biometrics
  - Experimental features found under 'About Phone" and settings.
  - **Version 8.1 Oreo**
  - Support for Neural networks API
  - **Version 7.1:**
  - Lots of developer features regarding storage, keyboard, etc.
  - **Version 7.0 (Nougat):**
  - File based encryption
  - **Version 6.0 Marshmallow:**
  - Fingerprint reader support
  - Post install permission requests
  - **Version 5.1 (Lollipop):**
  - Device protection. Owner must sign into Google account to unlock.

Android Q

- Android Q is the tenth major version of the Android operating system. It was first announced by Google on March 13, 2019. There have been two Beta versions, the second on April 3, 2019. Scheduled release September 2, 2019. It is said to have several new features:
- It was internally codenamed "Quince Tart“ Some sources say it was internally called “Queen Cake”
- New permissions to access location in background and to access photo, video and audio files
- Allows users to control when apps have permission to see their location: never, only when the app is in use (running), or all the time (when in the background).
- Better support for biometric authentication in apps.
- Support WPA3 Wifi Networks.
- Background apps can no longer jump into the foreground.
- Limited access to non-resettable device identifiers.
- Sharing shortcuts, which allow sharing content with a contact directly.
- Floating settings panel, that allow changing system settings directly from apps.
- Dynamic depth format for photos, which allow changing background blur after taking a photo.
- Support for the AV1 video codec, the HDR10+ video format and the Opus audio codec.
- A native MIDI API, allowing interaction with music controllers.

Android Q

- In February 2019, Google unveiled Adiantum, an encryption cipher designed primarily for use on devices that do not have hardware-accelerated support for the Advanced Encryption Standard (AES), such as low-end devices. This will be discussed in more detail in lesson 6.

- Device encryption is now mandatory on all Android 10 devices, regardless of specifications

- Android 10 supports WPA3 encryption protocol

What is Open Handset Alliance?

- "Open Handset Alliance™, a group of 47 technology and mobile companies have come together to accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience. Together we have developed Android™, the first complete, open, and free mobile platform. We are committed to commercially deploy handsets and services using the Android Platform."

Open Handset Alliance

Operators	Software Co.	Commercializat.	Semiconductor	Handset Manf
China Mobile	Ascender Corp.	Aplix	Audience	ACER
China Unicom	eBay	Noser Engineering	Broadcom Corp.	ASUS
KDDI Corp.	Esmertec	Astonishing Tribe	Intel Corp.	HTC
NTT DoCoMo	Google	Wind River Systems	Marvell Tech.	LG
Sprint Nextel	LivingImage	Omron Software	Group	Motorola
T-Mobile	NMS Comm.	...	Nvidia Corp.	Samsung
Telecom Italia	Nuance Comm.	Teleca	Qualcomm	ASUSTek
Telefónica	PacketVideo		SiRF Tech. Holdings	Garmin
Vodafone	SkyPop		Synaptics	Huawei Tech
Softbank	SONiVOX		Texas Instr.	LG
...	...		AKM Semicond.	Samsung
Ericsson	Borqs		ARM	...
			Atheros Comm	Sony Ericsson
			...	Toshiba
			EMP	

System facts

- As of 2018, Android targets versions 4.4, 4.9 or 4.14 of the Linux kernel
- Specifics depend on vendor
- File Structure
  - **/boot**
  - **/system**
  - **/recovery**
  - **/data**
  - **/cache**
  - **/misc**
  - **SD File card systems will also have**
  - **/sdcard**
  - **/sd-ext (not standard)**

/boot

- This is the boot partition of your Android device, as the name suggests.
- It includes the android kernel and the ramdisk.
- The device will not boot without this partition.

/system

- As the name suggests, this partition contains the entire Android OS, other than the kernel and the ramdisk.
- This includes the Android GUI and all the system applications that come pre-installed on the device.
- Wiping this partition will remove Android from the device without rendering it unbootable,
- and you will still be able to put the phone into recovery or bootloader mode to install a new ROM.

/recovery

- This is specially designed for backup.
- The recovery partition can be considered as an alternative boot partition,
- that lets the device boot into a recovery console for performing advanced recovery and maintenance operations on it.

/data

Again as the name suggest, it is called userdata partition.

This partition contains the user's data like your contacts, sms, settings and all android applications that you have installed.

While you perform factory reset on your device, this partition will be wiped out,

Then your device will be in the state, when you used for the first time or the way it was after the last official or custom ROM installation..

/cache

- I hope you have some idea about cache, as you are expert on internet browsing.
- This is the partition where Android stores frequently accessed data and app components.
- Wiping the cache doesn't effect your personal data but simply gets rid of the existing data there,
- which gets automatically rebuilt as you continue using the device.

/misc

- This partition contains miscellaneous system settings in form of on/off switches.
- These settings may include CID (Carrier or Region ID), USB configuration and certain hardware settings etc.
- This is an important partition and if it is corrupt or missing, several of the device's features will will not function normally.

System facts

Mobile devices SQLite databases

№	Type of data	Name of file
1	Phone book	\data\data\com.android.providers.contacts\ databases\contacts2.db
2	SMS, MMS messages	\data\data\com.android.providers.telephony\ databases\mmssms.db
3	Calendar	\data\com.android.providers.calendar\databases\ calendar.db
4	Log	\data\com.sec.android.provider.logsprovider\ databases\logs.db
5	User's data	\data\system\users\accounts.db
6	Web-browser history	\data\data\com.android.browser\databases\ browser2.db
7	Dictionary	\data\user\comc.android.providers.userdictionary\ databases\user_dict.db

From https://articles.forensicfocus.com/2014/10/28/extracting-data-from-dump-of-mobile-devices-running-android-operating-system/

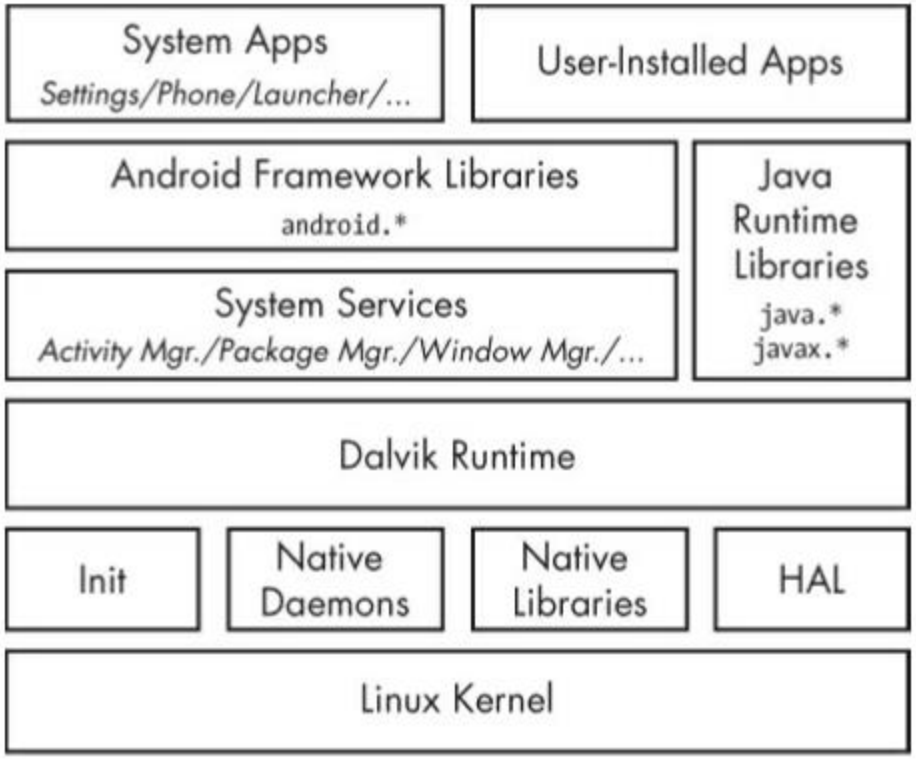
Storage Locations

- FOR SMS
- Samsung - nvm/sms/sms xxxx
- LG - sms/inbox.dat, sms/outbox.dat
- Motorola - nvm/seem/syn messg
- Motorola MMS - motorola/mms/xxxxxx.inb folder

Dates and Times

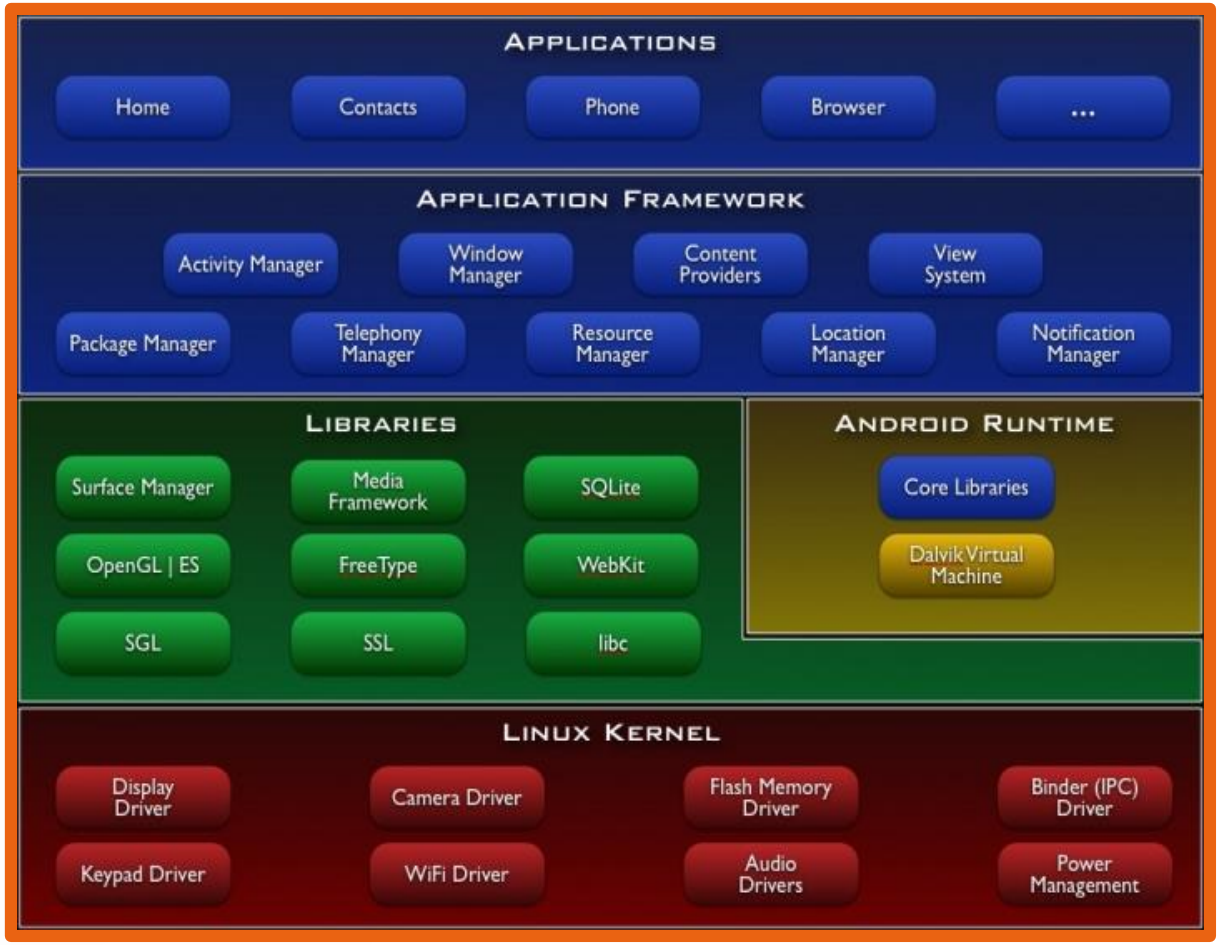
- UNIX, based on Jan 1, 1970
- GPS, based on Jan 6, 1980
- Some (such as Motorola) may use AOL, based on Jan 1, 1980

Android Architecture



-Elenkov, Nikolay. Android Security Internals: An In-Depth Guide to Android's Security Architecture

Android Architecture



Dalvik VM

- Until version 4.4 KitKat October 31, 2013 (After that it is the Android Runtime or ART).
- All applications written in Java and converted to the dalvik executable .dex
- Every android app runs its own process, with its own instance of the dalvik virtual machine
- Not a traditional JVM, but a custom VM designed to run multiple instances efficiently on a single device
- VM uses Linux kernel to handle low-level functionality incl. security, threading, process and memory management

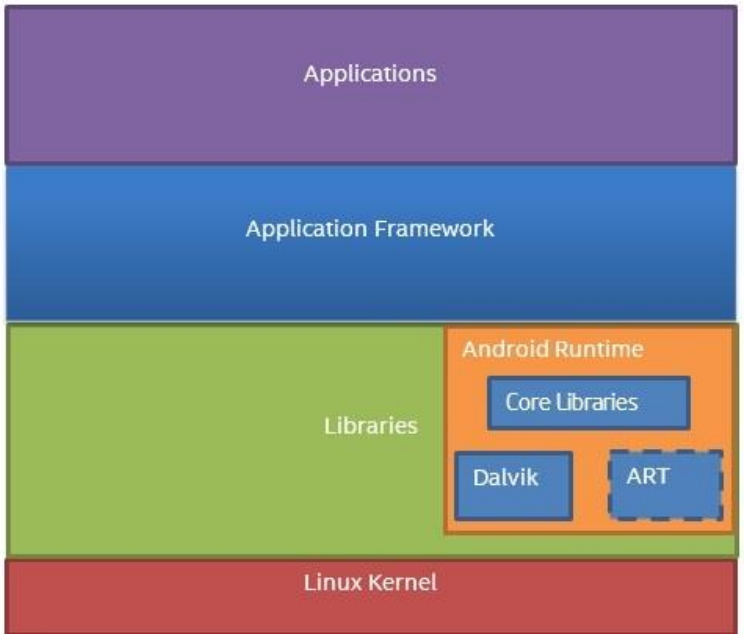
DVM

Dalvik is open-source software. Dan Bornstein originally wrote Dalvik VM which is responsible for running apps on Android devices.

- It is a Register based Virtual Machine.
- It is optimized for low memory requirements.
- It has been designed to allow multiple VM instances to run at once.
- Relies on the underlying OS for process isolation, memory management and threading support.
- Operates on DEX files.
- All hardware and system service access is managed using Dalvik (middle tier)

ART- Android Runtime

- To maintain backward compatibility, ART uses the same input bytecode as Dalvik, supplied through standard .dex files as part of APK files, while the .odex files are replaced with Executable and Linkable Format (ELF) executables



ART- Android Runtime

- ART reduces application execution overheads associated with Dalvik's interpretation and trace-based JIT compilation. However, ART requires additional time for the compilation when an application is installed. Furthermore, applications take up slightly larger amounts of secondary storage to store the compiled code

ART- Android Runtime

- ART performs the translation of the application's bytecode into native instructions that are later executed by the device's runtime environment. To maintain backward compatibility, ART uses the same input bytecode as Dalvik, supplied through standard .dex files as part of APK files, while the .odex files are replaced with Executable and Linkable Format (ELF) executables. Once an application is compiled by using ART's on-device dex2oat utility, it is run solely from the compiled ELF executable; as a result, ART eliminates various application execution overheads associated with Dalvik's interpretation and trace-based JIT compilation.

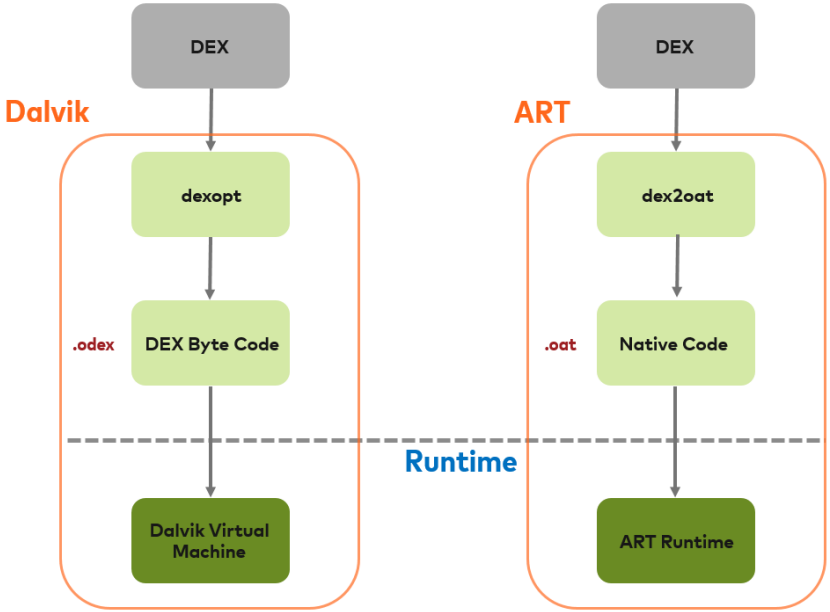
Dalvik vs ART

Dalvik

- Just In Time (JIT) Compiling

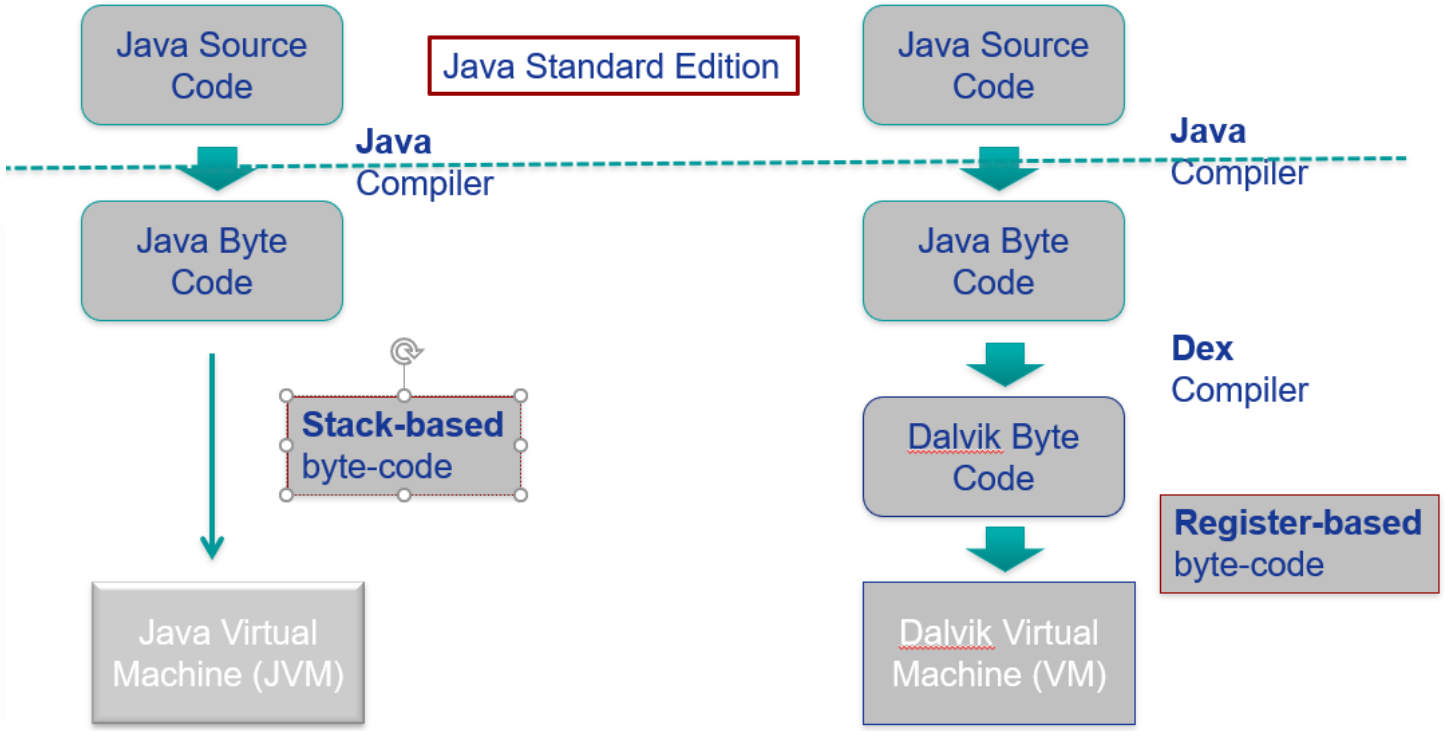
ART

- Ahead of Time(AOT) Compiling



Dalvik vs ART

Android VM



Luca Bedogni, Marco Di Felice - Programming with Android – System Architecture

Android VM

- Dalvik and Oracle's JVM have different architectures — register-based in Dalvik versus stack-based in the JVM — and different instruction sets. Let's look at a simple example to illustrate the differences between the two VMs

Application Fundamentals

- Activities
  - application presentation layer
- Services
  - invisible components, update data sources, visible activities, trigger notifications
  - perform regular processing even when app is not active or invisible
- Content Providers
  - shareable data store
- Intents
  - message passing framework
  - broadcase messages system wide, for an action to be performed
- Broadcast receivers
  - consume intent broadcasts
  - lets app listen for intents matching a specific criteria like location
- Notifications
  - Toast notification
  - Status Bar Notification
  - Dialog notification

Android S/W Stack – App Framework

Feature	Role
View System	Used to build an application, including lists, grids, text boxes, buttons, and embedded web browser
Content Provider	Enabling applications to access data from other applications or to share their own data
Resource Manager	Providing access to non-code resources (localized string , graphics, and layout files)
Notification Manager	Enabling all applications to display customer alerts in the status bar
Activity Manager	Managing the lifecycle of applications and providing a common navigation backstack

Applications

- All apps (native and 3rd party) are written using the same APIs and run on the same run time executable
- All apps have APIs for hardware access, location-based services, support for background services, map-based activities, 2D and 3D graphics.
- App Widgets are miniature app views that can be embedded in other apps like Home Screen

App Priority and Processes

- Android apps do not have control over their own life cycles
- Aggressively manages resources to ensure device responsiveness and kills process/apps when needed

- Active Process – critical priority
- Visible Process – high priority
- Started Service Process
- Background Process – low priority
- Empty process

Client apps

- Developed using the Android SDK and installed on user devices
- Compiled Java code, with data and resource – bundled by Android Asset Packaging tool (AAPT) into Android package or .apk
- All applications have Android Manifest file in its root directory
  - provides essential information about app
- Could be installed directly on phone, but necessary to be distributed thru Market

Web Apps

- An alternative to standalone apps
- Developed using web standards and accessed through browser – nothing to install on devices
- Mixing client and web apps is also possible – Client apps can embed web pages using "Webview" in Android app

SDK

- Android APIs, Full Documentation and Sample code
- Development tools
  - Dalvik Debug Monitor Service (DDMS)
  - Android Debug Bridge (ADB)
  - Android Emulator
- Online support and blog
- Native Development Kit also available
  - allows developers to implement parts of apps in native-code languages like C/C++
  - Plug in available to use Eclipse integrated development environment
- Developer forums and developer phones from Google, MOTO Dev studio from Motorola

Tools

- Phone
- Eclipse (http://www.eclipse.org/downloads/)
  - Android Plugin (ADT)
- Android SDK (http://developer.android.com/sdk/index.html)
  - Install everything except Additional SDK Platforms, unless you want to
- Windows Users: may need to install Motorola Driver directly (http://www.motorola.com/Support/US-EN/Support-Homepage/Software_and_Drivers/USB-and-PC-Charging-Drivers)

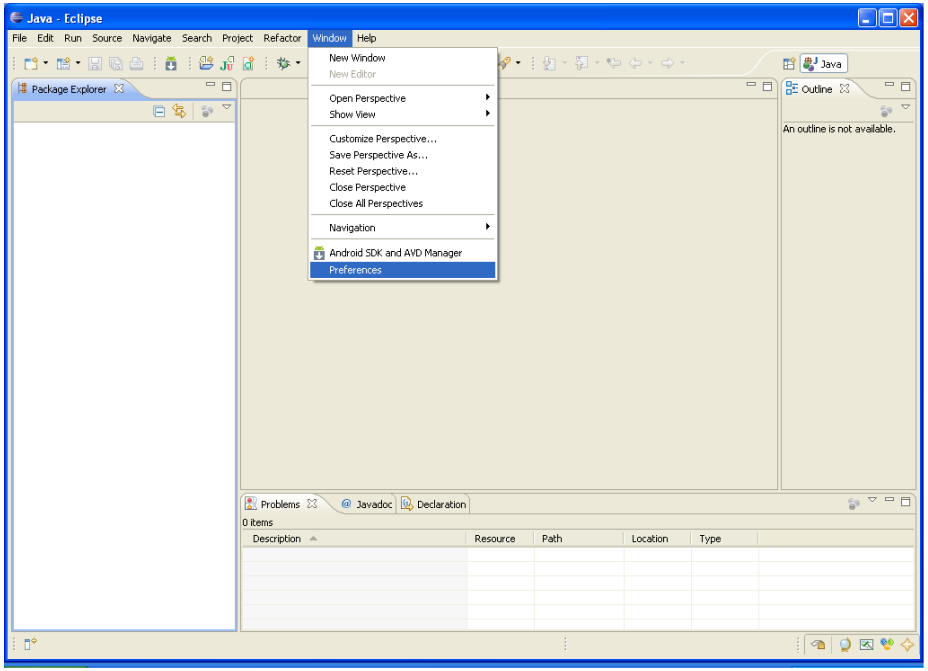
Android SDK

- Once installed open the SDK Manager
- Install the desired packages
- Create an Android Virtual Device (AVD)

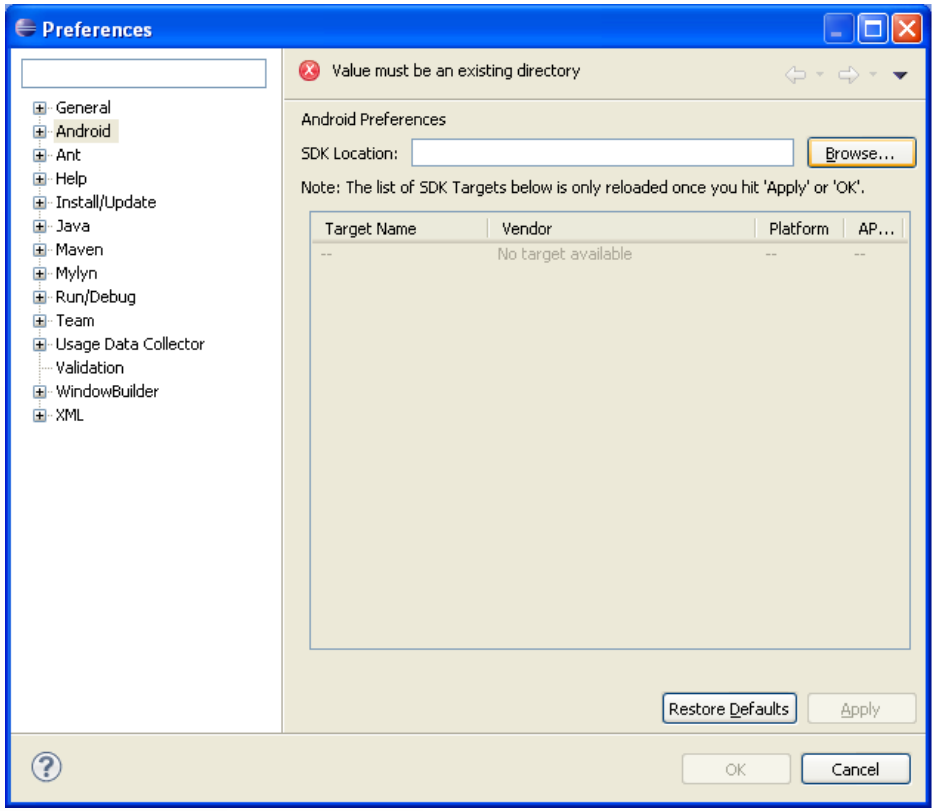
ADT Plugin for Eclipse

- In Eclipse, go to Help -> Install New Software
- Click ‘Add’ in top right
- Enter:
  - Name: ADT Plugin
  - Location: https://dl-ssl.google.com/android/eclipse/
- Click OK, then select ‘Developer Tools’, click Next
- Click Next and then Finish
- Afterwards, restart Eclipse
- Specify SDK location (next 3 slides)
  - Must do this every time start a new project in a new location (at least in Windows)

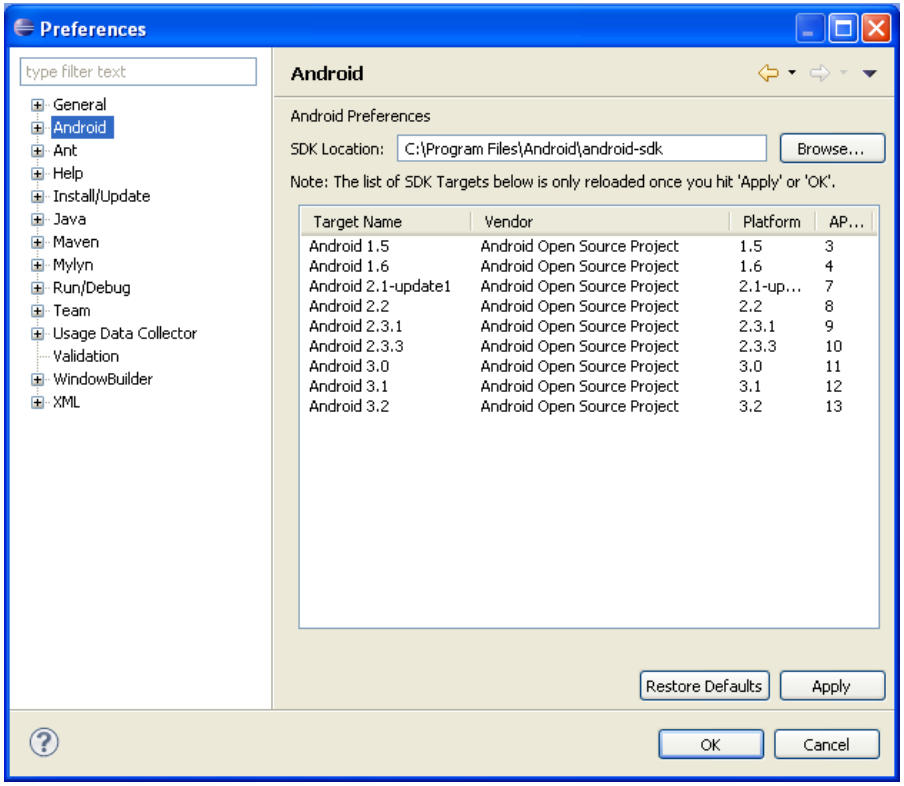
ADT Plugin for Eclipse



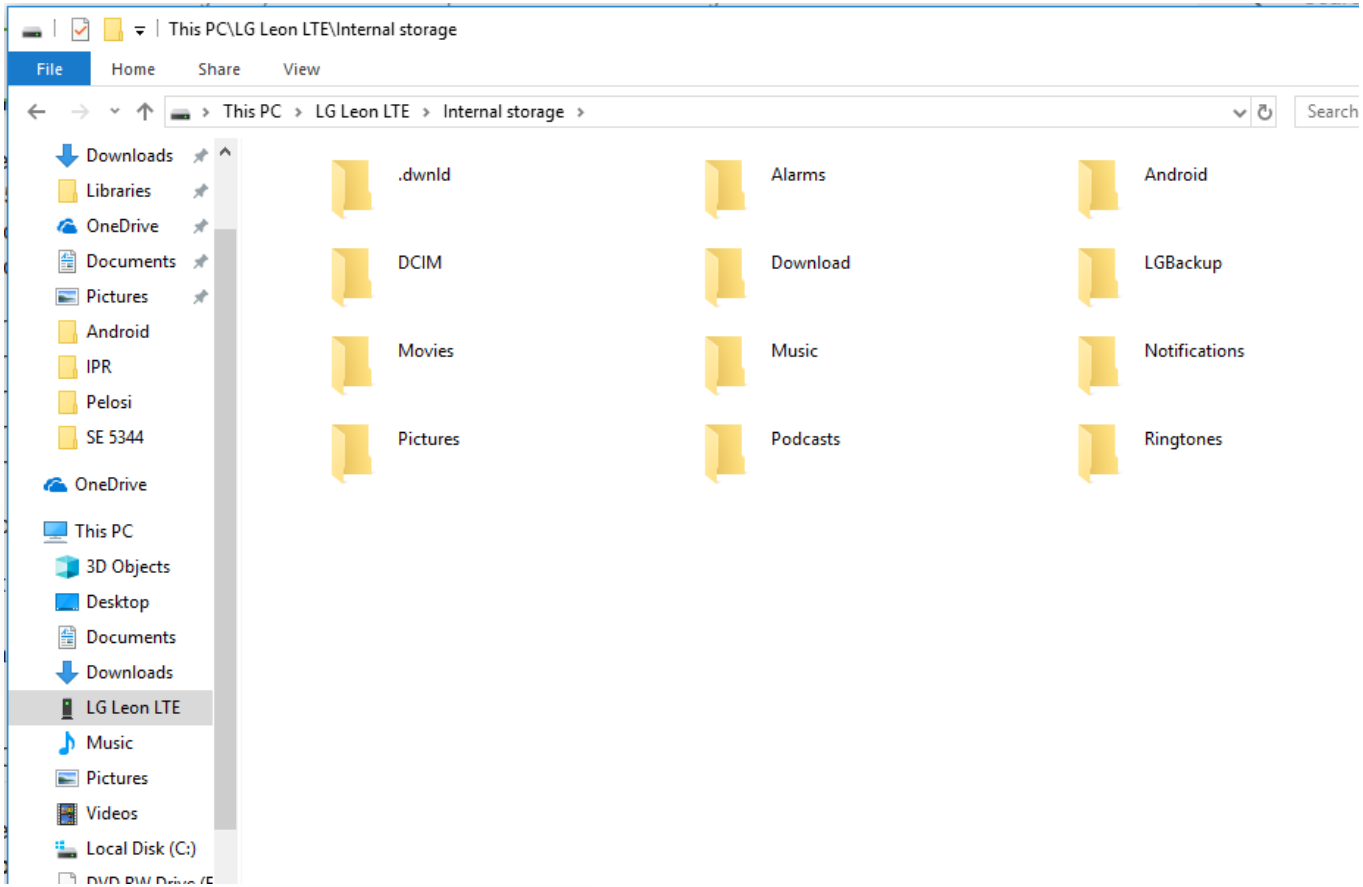
ADT Plugin for Eclipse



ADT Plugin for Eclipse



Using ADB First connect the device



ADB

Note: the following three slides are from https://developer.android.com/studio/command-line/adb

- Android Debug Bridge (adb) is a versatile command-line tool that lets you communicate with a device. The adb command facilitates a variety of device actions, such as installing and debugging apps, and it provides access to a Unix shell that you can use to run a variety of commands on a device. It is a client-server program that includes three components:

- A client, which sends commands. The client runs on your development machine. You can invoke a client from a command-line terminal by issuing an adb command.
- A daemon (adbd), which runs commands on a device. The daemon runs as a background process on each device.
- A server, which manages communication between the client and the daemon. The server runs as a background process on your development machine.

Developer Mode

- Androids have to be in developer mode to extract data from them.
  - **Developer Options on Gingerbread (Android 2.3):**
  - *Settings> Applications> Development> USB Debugging*
  - **Developer Options on ICS (Android 4.0):**
  - *Settings> Developer Options> USB Debugging*
  - **Developer Options on JB (Android 4.1):**
  - *Settings> Developer Options> USB Debugging*

Developer Mode

- Open ***Settings> About*** on your Android phone or tablet.
- If you have a **Samsung Galaxy S4, Note 8.0, Tab 3** or any other Galaxy device with Android 4.2, open ***Settings> More tab> About*** and tap it.
- If you have Galaxy Note 3 or any Galaxy device with Android 4.3, go to **Galaxy Note 3** from ***Settings> General> About*** and tap the Build version 7 times.
- Now scroll to *Build number* and tap it **7** times.
- After tapping the Build Number **7** times, you will see a message "*You are now a developer!*" If you have a Galaxy S4 or any other Samsung Galaxy device with Android 4.2, the message reads as follows- "*Developer mode has been enabled*".

Developer Mode

- Return to the main Settings menu and now you'll be able to see **Developer Options**.
- Tap on Developer options and mark the box in front of **USB Debugging** to enable it.
- To **disable USB Debugging** mode later, you can uncheck the box before the option
- To enable **Developer Options**, go to ***Settings>Developer options*** and tap on the **ON/OFF** slider on the top of the page.

ADB

- When you start an adb client, the client first checks whether there is an adb server process already running. If there isn't, it starts the server process. When the server starts, it binds to local TCP port 5037 and listens for commands sent from adb clients—all adb clients use port 5037 to communicate with the adb server.
- The server then sets up connections to all running devices. It locates emulators by scanning odd-numbered ports in the range 5555 to 5585, the range used by the first 16 emulators. Where the server finds an adb daemon (adbd), it sets up a connection to that port. Note that each emulator uses a pair of sequential ports — an even-numbered port for console connections and an odd-numbered port for adb connections. For example:
- Emulator 1, console: 5554
- Emulator 1, adb: 5555
- Emulator 2, console: 5556
- Emulator 2, adb: 5557
- As shown, the emulator connected to adb on port 5555 is the same as the emulator whose console listens on port 5554.
- Once the server has set up connections to all devices, you can use adb commands to access those devices. Because the server manages connections to devices and handles commands from multiple adb clients, you can control any device from any client (or from a script).

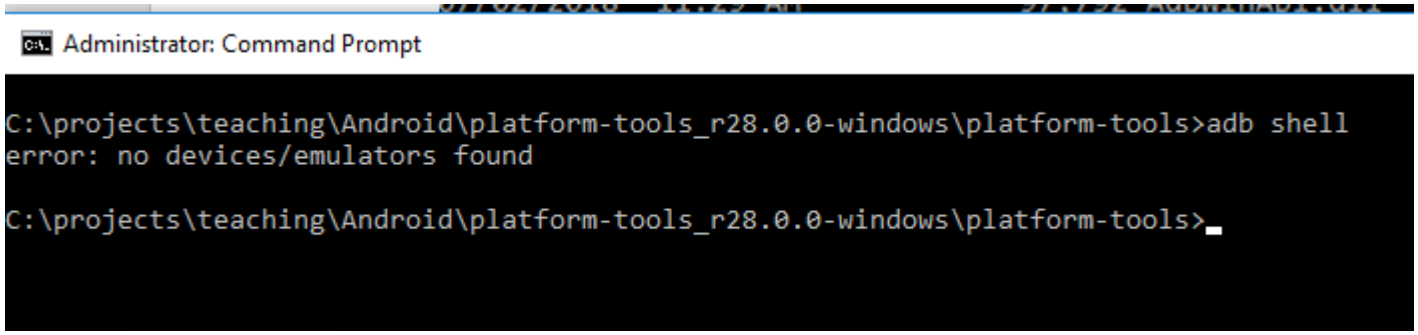
ADB

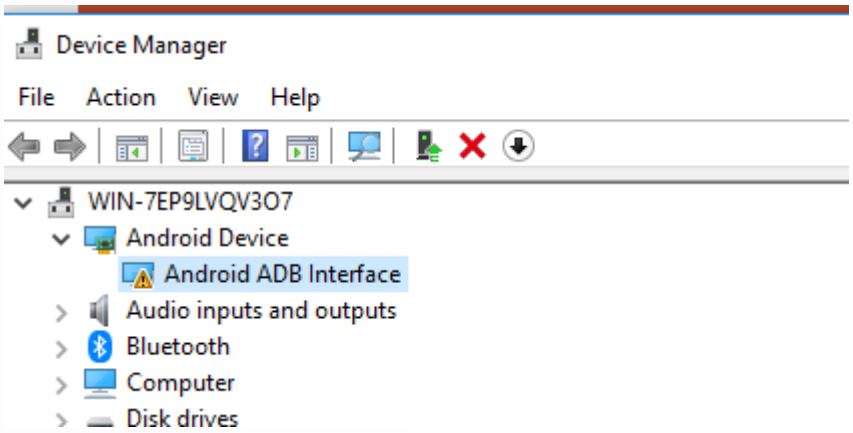
- To use adb with a device connected over USB, you must enable USB debugging in the device system settings, under Developer options.
- On Android 4.2 and higher, the Developer options screen is hidden by default. To make it visible, go to Settings > About phone and tap Build number seven times. Return to the previous screen to find Developer options at the bottom.
- On some devices, the Developer options screen might be located or named differently.
- You can now connect your device with USB. You can verify that your device is connected by executing adb devices from the android_sdk/platform-tools/ directory. If connected, you'll see the device name listed as a "device."

ADB

```
C:\projects\teaching\Android\platform-tools_r28.0.0-windows\platform-tools>dir
 Volume in drive C has no label.
 Volume Serial Number is B42C-686E

 Directory of C:\projects\teaching\Android\platform-tools_r28.0.0-windows\platform-tools

07/02/2018  11:29 AM    <DIR>          .
07/02/2018  11:29 AM    <DIR>          ..
07/02/2018  11:29 AM         1,850,368 adb.exe
07/02/2018  11:29 AM            97,792 AdbWinApi.dll
07/02/2018  11:29 AM            62,976 AdbWinUsbApi.dll
07/02/2018  11:29 AM    <DIR>          api
07/02/2018  11:29 AM           145,920 dmtracedump.exe
07/02/2018  11:29 AM           333,824 etc1tool.exe
07/02/2018  11:29 AM           857,088 fastboot.exe
07/02/2018  11:29 AM            43,008 hprof-conv.exe
07/02/2018  11:29 AM    <DIR>          lib64
07/02/2018  11:29 AM           210,625 libwinpthread-1.dll
07/02/2018  11:29 AM           346,112 make_f2fs.exe
07/02/2018  11:29 AM             1,178 mke2fs.conf
07/02/2018  11:29 AM         1,041,920 mke2fs.exe
07/02/2018  11:29 AM           335,729 NOTICE.txt
07/02/2018  11:29 AM                38 source.properties
07/02/2018  11:29 AM           829,440 sqlite3.exe
07/02/2018  11:29 AM    <DIR>          systrace
              14 File(s)      6,156,018 bytes
               5 Dir(s)  209,355,530,240 bytes free

C:\projects\teaching\Android\platform-tools_r28.0.0-windows\platform-tools>
```


ADB


Administrator: Command Prompt
C:\projects\teaching\Android\platform-tools_r28.0.0-windows\platform-tools>adb shell
error: no devices/emulators found
C:\projects\teaching\Android\platform-tools_r28.0.0-windows\platform-tools>

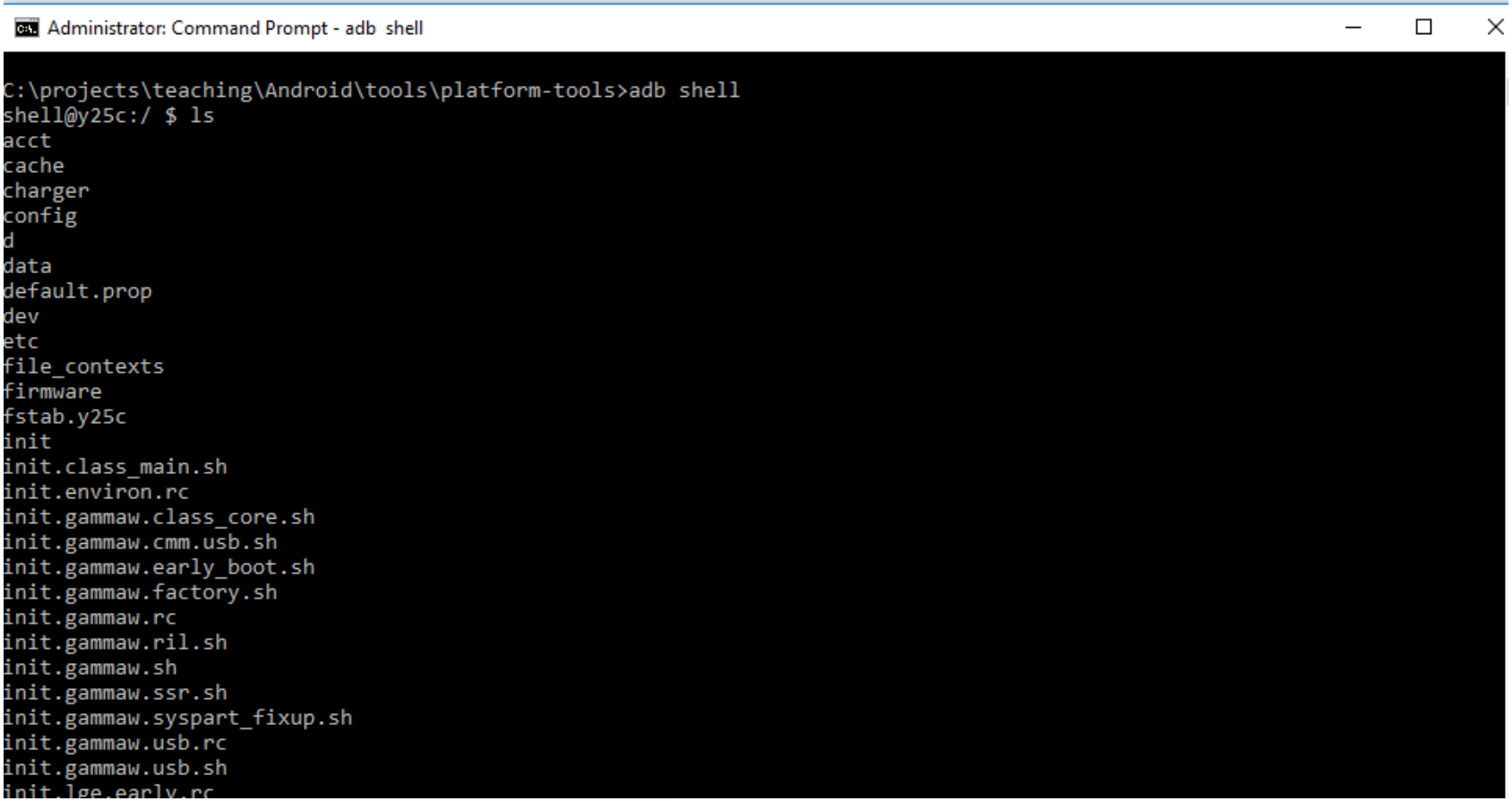

Device Manager
File
Action
View
Help
WIN-7EP9LVQV3O7
Android Device
Android ADB Interface
Audio inputs and outputs
Bluetooth
Computer
Disk drives

ADB

```
C:\projects\teaching\Android\platform-tools_r28.0.0-windows\platform-tools>adb devices
List of devices attached


C:\projects\teaching\Android\platform-tools_r28.0.0-windows\platform-tools>
```

```
C:\projects\teaching\Android\tools\platform-tools>adb shell
error: no devices/emulators found

C:\projects\teaching\Android\tools\platform-tools>adb shell
shell@y25c:/ $ exit

C:\projects\teaching\Android\tools\platform-tools>adb devices
List of devices attached
LGL16C9f11b9bf  device
```


ADB Shell


Administrator: Command Prompt - adb shell
C:\projects\teaching\Android\tools\platform-tools>adb shell
shell@y25c:/ $ ls
acct
cache
charger
config
d
data
default.prop
dev
etc
file_contexts
firmware
fstab.y25c
init
init.class_main.sh
init.environ.rc
init.gammaw.class_core.sh
init.gammaw.cmm.usb.sh
init.gammaw.early_boot.sh
init.gammaw.factory.sh
init.gammaw.rc
init.gammaw.ril.sh
init.gammaw.sh
init.gammaw.ssr.sh
init.gammaw.syspart_fixup.sh
init.gammaw.usb.rc
init.gammaw.usb.sh
init.lge.early.rc

Shell commands

- http://adbshell.com/commands

ADB basic commands

// launch adb and see available commands

adb

// start a shell this well let you issue Linux shell commands!

- adb shell

// if you have gestures on the phone and you want to

// copy them to the test directory on your system

- adb pull /sdcard/gestures ~/test

//uninstall an application (Not something you normally

// do in a forensic exam

- adb uninstall <packagename>

//lists you all application in the order of their memory
//consumption.

- adb shell procrank

ADB basic commands

// screen capture

- adb shell screencap /sdcard/screen.png

// screen recording

- adb shell screen record /sdcard/demo.mp4

// backup and restore the device

- adb backup
- adb restore <archive name>

// you can reboot, and if you want reboot into recovery or bootloader mode.//

- adb reboot
- adb reboot recovery
- adb reboot bootloader

// see the partitions

/proc/partitions

ADB advanced commands

// Forward port

- adb forward <local> <remote>
- adb forward tcp:8000 tcp:9000

// logs

- adb logcat
  - adb logcat *:D filter to only show Debug level
  - adb logcat *:W filter to only show Warning level
  - adb logcat *:F filter to only show Fatal level
  - adb logcat -b event View the buffer containing events-related messages.

// Remounts file system with read/write access

- adb remount

// Screen resolution

- adb shell wm size
- adb shell wm size WxH

Common Linux commands

// List open files and what process opened them

- lsof
  - lsof -u [user-name]
  - lsof -i 4 ipv4
  - lsof -i 6 ipv6
  - lsof -p [PID]

//List all processes in a tree

- pstree

// command history

- History

Current directory

- pwd

Common Linux commands

// list contents of a folder

- ls
  - *ls -l* shows file or directory, size, modified date and time, file or folder name and owner of file and its permission.
  - ls- F lists directories with a / at the end
  - ls – a all files including hidden tiles
  - ls -R list recursively directory tree

Common Linux commands

//List all processes based on resource use

- Top
- top -u username
  - PID: Shows task's unique process id.
  - PR: Stands for priority of the task.
  - SHR: Represents the amount of shared memory used by a task.
  - VIRT: Total virtual memory used by the task.
  - USER: User name of owner of task.
  - %CPU: Represents the CPU usage.
  - TIME+: CPU Time, the same as 'TIME', but reflecting more granularity through hundredths of a second.
  - SHR: Represents the Shared Memory size (kb) used by a task.
  - NI: Represents a Nice Value of task. A Negative nice value implies higher priority, and positive Nice value means lower priority.
  - %MEM: Shows the Memory usage of task

Common Linux commands

// show processes

- ps
  - ps –e or ps – A will list all processes
  - For full format listing add –f such as ps –ef
  - All processes running as root ps -U root -u root

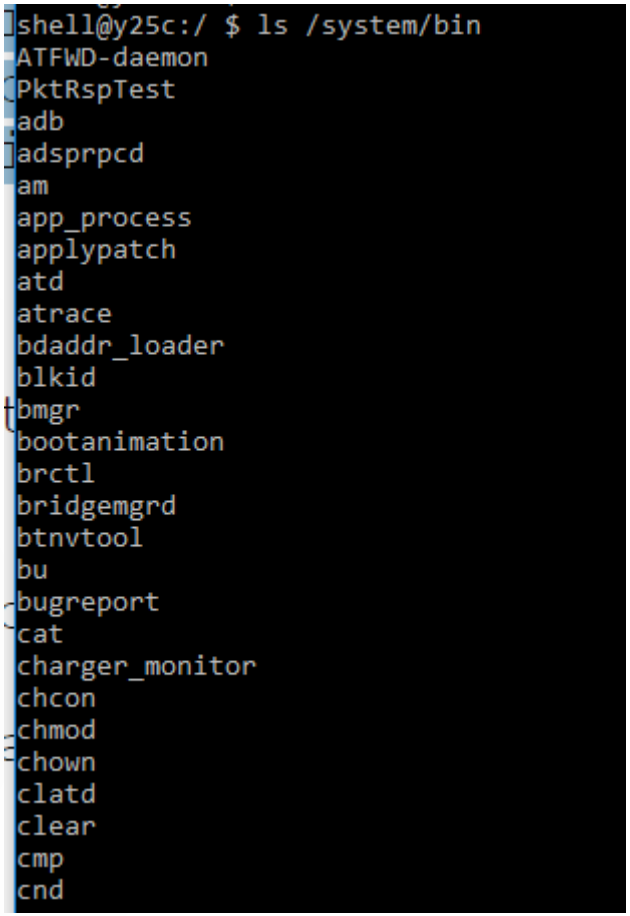
//network status

- Netstat
- netstat –s statistics by protocol
- netstat –tp service with processor id

//Find out what a file is regardless of extension

- file

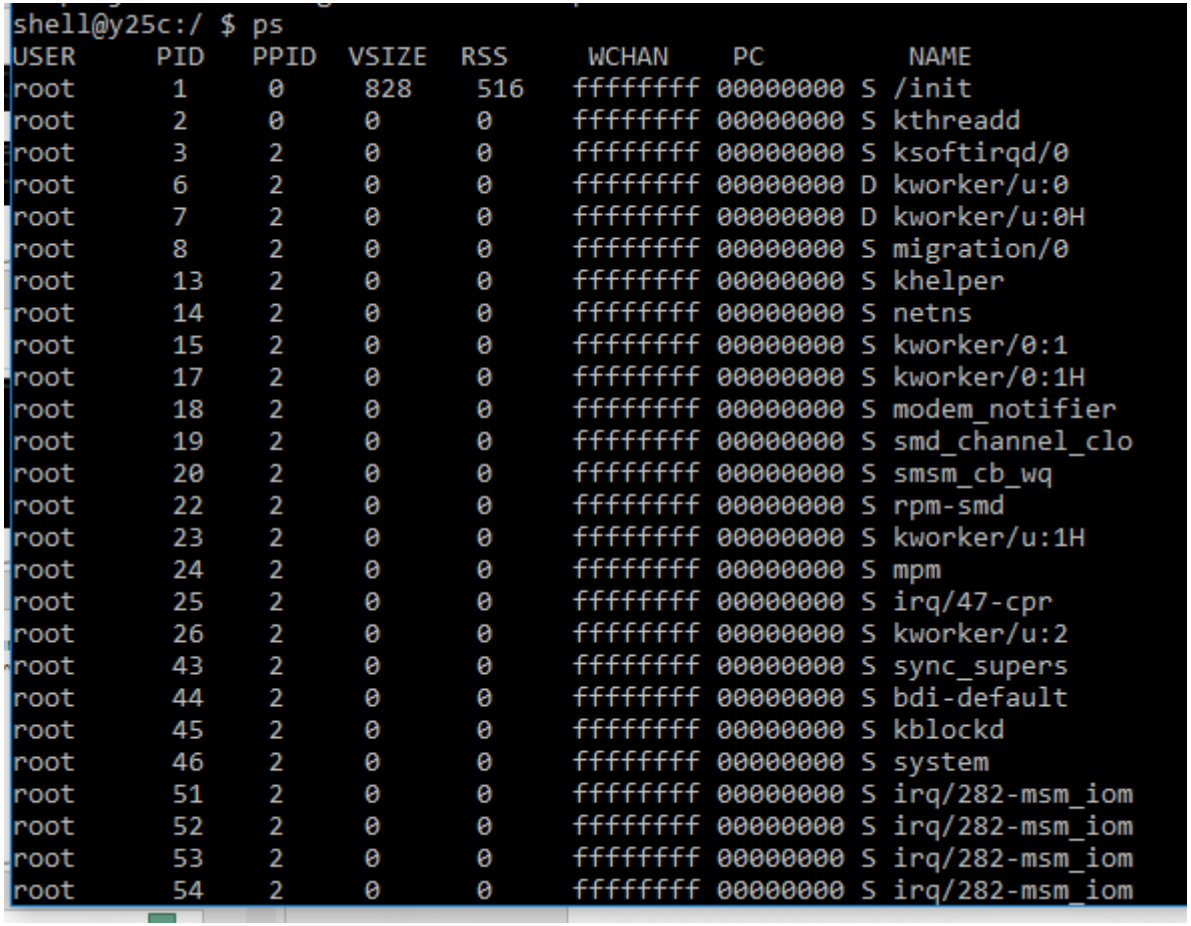
Find the Linux commands



Dumpsys activity

```
[2J[Hshell@y25c:/ $ dumpsys activity
ACTIVITY MANAGER PENDING INTENTS (dumpsys activity intents)
  * PendingIntentRecord{422beb08 com.android.vending startService}
  * PendingIntentRecord{424b5250 com.android.mms broadcastIntent}
  * PendingIntentRecord{422fa5c8 com.android.settings broadcastIntent}
  * PendingIntentRecord{42321b48 android startActivity}
  * PendingIntentRecord{4257f428 android startActivity}
  * PendingIntentRecord{422880d8 com.google.android.gms startService}
  * PendingIntentRecord{421e1c38 android broadcastIntent}
  * PendingIntentRecord{41f42ab8 com.wsandroid.suite.lge broadcastIntent}
  * PendingIntentRecord{425530d8 com.google.android.googlequicksearchbox startService}
  * PendingIntentRecord{423e6b80 com.google.android.gms broadcastIntent}
  * PendingIntentRecord{424bc358 com.google.android.gms broadcastIntent}
  * PendingIntentRecord{422f80d0 com.android.settings broadcastIntent}
  * PendingIntentRecord{42455c38 android broadcastIntent}
  * PendingIntentRecord{424ce5a8 com.google.android.talk broadcastIntent}
  * PendingIntentRecord{41dc86c8 android startActivity}
  * PendingIntentRecord{42297050 com.google.android.gms broadcastIntent}
  * PendingIntentRecord{4234fa80 android broadcastIntent}
  * PendingIntentRecord{4255f210 com.google.android.gms broadcastIntent}
  * PendingIntentRecord{4255f6c0 com.google.android.gms broadcastIntent}
  * PendingIntentRecord{425451c8 com.google.android.gms broadcastIntent}
  * PendingIntentRecord{42358b98 com.lge.clock broadcastIntent}
  * PendingIntentRecord{424a4928 com.android.phone broadcastIntent}
  * PendingIntentRecord{42297530 com.google.android.gms broadcastIntent}
  * PendingIntentRecord{42462a98 com.google.android.gms broadcastIntent}
  * PendingIntentRecord{4255f450 com.google.android.gms broadcastIntent}
  * PendingIntentRecord{4237cfb8 com.android.settings broadcastIntent}
  * PendingIntentRecord{41f412b0 com.google.android.gms broadcastIntent}
  * PendingIntentRecord{423bc0a0 com.lge.appbox.client startService}
```


ps



```
shell@y25c:/ $ ps
USER      PID   PPID  VSIZE  RSS     WCHAN    PC         NAME
root      1     0     828    516   ffffffff 00000000 S /init
root      2     0     0      0     ffffffff 00000000 S kthreadd
root      3     2     0      0     ffffffff 00000000 S ksoftirqd/0
root      6     2     0      0     ffffffff 00000000 D kworker/u:0
root      7     2     0      0     ffffffff 00000000 D kworker/u:0H
root      8     2     0      0     ffffffff 00000000 S migration/0
root      13    2     0      0     ffffffff 00000000 S khelper
root      14    2     0      0     ffffffff 00000000 S netns
root      15    2     0      0     ffffffff 00000000 S kworker/0:1
root      17    2     0      0     ffffffff 00000000 S kworker/0:1H
root      18    2     0      0     ffffffff 00000000 S modem_notifier
root      19    2     0      0     ffffffff 00000000 S smd_channel_clo
root      20    2     0      0     ffffffff 00000000 S smsm_cb_wq
root      22    2     0      0     ffffffff 00000000 S rpm-smd
root      23    2     0      0     ffffffff 00000000 S kworker/u:1H
root      24    2     0      0     ffffffff 00000000 S mpm
root      25    2     0      0     ffffffff 00000000 S irq/47-cpr
root      26    2     0      0     ffffffff 00000000 S kworker/u:2
root      43    2     0      0     ffffffff 00000000 S sync_supers
root      44    2     0      0     ffffffff 00000000 S bdi-default
root      45    2     0      0     ffffffff 00000000 S kblockd
root      46    2     0      0     ffffffff 00000000 S system
root      51    2     0      0     ffffffff 00000000 S irq/282-msm_iom
root      52    2     0      0     ffffffff 00000000 S irq/282-msm_iom
root      53    2     0      0     ffffffff 00000000 S irq/282-msm_iom
root      54    2     0      0     ffffffff 00000000 S irq/282-msm_iom
```

Pm list packages

```
shell@y25c:/ $ pm list packages
package:com.lge.shutdownmonitor
package:com.qualcomm.timeservice
package:com.qualcomm.atfwd
package:com.android.defcontainer
package:com.lge.camera
package:com.android.providers.partnerbookmarks
package:com.android.settingsaccessibility
package:com.android.contacts
package:com.lge.eulaprovider
package:com.android.phone
package:com.lge.lginstallservies
package:com.lge.launcher2
package:com.android.calculator2
package:com.lge.mtalk.voicecommand
package:com.android.htmlviewer
package:com.lge.qmemoplus
package:com.google.android.gsf.login
package:com.lge.lockscreensettings
package:com.android.bluetooth
package:com.android.providers.calendar
package:com.lge.updatecenter
package:com.lge.ime
package:com.android.calendar
package:com.lge.filemanager
package:com.android.browser
package:com.lge.sui.widget
package:com.lge.gnss.airtest
```

Pm list packages

pm list packages -f See their associated file
pm list packages -d Filter to only show disabled packages
pm list packages -e Filter to only show enabled packages
pm list packages -i See the installer for the packages
pm list packages -u Also include uninstalled packages

netstat

```
C:\projects\teaching\Android\tools\platform-tools>adb shell
shell@y25c:/ $ netstat
Proto Recv-Q Send-Q Local Address          Foreign Address        State
 tcp       0      0 127.0.0.1:5038         0.0.0.0:*              LISTEN
 tcp       0      0 127.0.0.1:56003        127.0.0.1:5038         TIME_WAIT
 tcp       0      0 127.0.0.1:5038         127.0.0.1:56002        TIME_WAIT
shell@y25c:/ $ _
```


netstat

```
C:\projects\teaching\Android\tools\platform-tools>adb shell
shell@y25c:/ $ netstat
Proto Recv-Q Send-Q Local Address          Foreign Address        State
 tcp       0      0 127.0.0.1:5038         0.0.0.0:*              LISTEN
 tcp       0      0 127.0.0.1:56003        127.0.0.1:5038         TIME_WAIT
 tcp       0      0 127.0.0.1:5038         127.0.0.1:56002        TIME_WAIT
shell@y25c:/ $ _
```


dumpstate

- adb shell dumpstate
- adb shell dumpstate > state.logs

```
rec[578]: time=07-03 10:59:27.662 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[579]: time=07-03 10:59:30.669 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[580]: time=07-03 10:59:33.675 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[581]: time=07-03 10:59:36.682 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[582]: time=07-03 10:59:39.689 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[583]: time=07-03 10:59:42.696 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[584]: time=07-03 10:59:45.703 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[585]: time=07-03 10:59:48.709 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[586]: time=07-03 10:59:51.717 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[587]: time=07-03 10:59:54.724 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[588]: time=07-03 10:59:57.731 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[589]: time=07-03 11:00:00.738 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[590]: time=07-03 11:00:03.745 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[591]: time=07-03 11:00:06.752 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[592]: time=07-03 11:00:09.760 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[593]: time=07-03 11:00:12.772 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[594]: time=07-03 11:00:15.780 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[595]: time=07-03 11:00:18.788 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[596]: time=07-03 11:00:21.793 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[597]: time=07-03 11:00:24.798 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[598]: time=07-03 11:00:27.805 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[599]: time=07-03 11:00:30.811 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[600]: time=07-03 11:00:33.817 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[601]: time=07-03 11:00:36.824 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[602]: time=07-03 11:00:39.831 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[603]: time=07-03 11:00:42.838 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[604]: time=07-03 11:00:45.852 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[605]: time=07-03 11:00:48.865 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[606]: time=07-03 11:00:51.880 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
rec[607]: time=07-03 11:00:54.895 processed=L2ConnectedState org=ConnectedState dest=<null> what=131155(0x20053)
```

View Partitions

```
127|shell@y25c:/ $ cat /proc/partitions
major minor  #blocks  name

 253        0     104260 zram0
 179        0    3817472 mmcblk0
 179        1      65536 mmcblk0p1
 179        2       1024 mmcblk0p2
 179        3        512 mmcblk0p3
 179        4        512 mmcblk0p4
 179        5        512 mmcblk0p5
 179        6       2048 mmcblk0p6
 179        7        512 mmcblk0p7
 179        8        512 mmcblk0p8
 179        9       2048 mmcblk0p9
 179       10       2048 mmcblk0p10
 179       11       3072 mmcblk0p11
 179       12       3072 mmcblk0p12
 179       13      16384 mmcblk0p13
 179       14      32768 mmcblk0p14
 179       15      22528 mmcblk0p15
 179       16      22528 mmcblk0p16
 179       17      22528 mmcblk0p17
 179       18       3072 mmcblk0p18
 179       19        512 mmcblk0p19
 179       20        512 mmcblk0p20
 179       21        512 mmcblk0p21
```

Other commands

- adb sideload <update.zip> flashing/restoring Android update.zip packages.
- adb root restarts the adbd daemon with root permissions
- Take a screen shot
- adb shell screencap /sdcard/screen.png
- Download it
- adb pull /sdcard/screen.png
- Upload something
- adb push myMusicFile.mp4 /sdcard/music
- Dump sdcard to text file
- dumpstate -o /sdcard/myDump.txt

Other commands

- shell data
- getprop ro.product.model
- getprop ro.build.version.release
- getprop ro.serialno
- am start -n com.android.settings/com.android.settings.Settings
- uninstall PACKAGE_NAME
- install -r APK_FILE
- screencap -p | perl -pe 's/\x0D\x0A/\x0A/g' > screen.png
- adb reboot
- adb install -r myApplication.apk

Using am

- The activity manager

```
C:\projects\teaching\Android\tools\platform-tools>adb shell
shell@y25c:/ $ am
usage: am [subcommand] [options]
usage: am start [-D] [-W] [-P <FILE>] [--start-profiler <FILE>]
               [--R COUNT] [-S] [--opengl-trace]
               [--user <USER_ID> | current] <INTENT>
       am startservice [--user <USER_ID> | current] <INTENT>
       am stopservice [--user <USER_ID> | current] <INTENT>
       am force-stop [--user <USER_ID> | all | current] <PACKAGE>
       am kill [--user <USER_ID> | all | current] <PACKAGE>
       am kill-all
       am broadcast [--user <USER_ID> | all | current] <INTENT>
       am instrument [-r] [-e <NAME> <VALUE>] [-p <FILE>] [-w]
               [--user <USER_ID> | current]
               [--no-window-animation] <COMPONENT>
       am profile start [--user <USER_ID> current] <PROCESS> <FILE>
       am profile stop [--user <USER_ID> current] [<PROCESS>]
       am dumpheap [--user <USER_ID> current] [-n] <PROCESS> <FILE>
       am set-debug-app [-w] [--persistent] <PACKAGE>
       am clear-debug-app
       am monitor [--gdb <port>]
       am hang [--allow-restart]
       am restart
       am idle-maintenance
       am screen-compat [on|off] <PACKAGE>
       am to-uri [INTENT]
       am to-intent-uri [INTENT]
       am switch-user <USER_ID>
       am stop-user <USER_ID>
```

Other commands

- am start -n com.someapp.myapplication/.MainActivity
- am startservice <service>
- am kill <PACKAGE>
- am kill-all
- am start -n com.android.settings/.wifi.WifiSettings - opens WiFi.
- am start -n com.android.settings/.wifi.WifiInfo - fetches WiFi info.
- am start -n com.android.settings/.wifi.WifiStatusTest - shows WiFi status.
- am start -n com.android.settings/.LanguageSettings - opens language settings.
- am start -n com.android.settings/.DevelopmentSettings - opens development

Extracting data with dd

- Since Android is based on Linux, if you wish to copy the entire storage, or just part, you can do it with dd, but you have to connect to it as you normally would (i.e. plug cable into your computer)
- **dd** is a common Unix/Linlux program whose primary purpose is the low-level copying and conversion of raw data
- The name is an allusion to mainframe JCL DD statement.
- It is jokingly said to stand for "disk destroyer", "data destroyer", or "delete data", since, being used for low-level operations on hard disks, a small mistake, such as reversing the **if** and **of** parameters, can possibly result in the loss of some or all data on a disk

Extracting data with dd

We use the dd command to read the first partition:

dd if=/dev/hda1 | nc 192.168.0.2 8888 –w 3

. Other examples:

dd if =/ dev/ block/ mmcblk0 of =/ sdcard/ blk0. img
bs = 4096 conv = notrunc, noerror, sync

Dcfldd

dcfldd is an enhanced version of dd developed by the U.S. Department of Defense Computer Forensics Lab. Features include:

- On-the-fly hashing of the transmitted data.
- Progress bar of how much data has already been sent.
- Wiping of disks with known patterns.
- Bit by bit image verification.
- The output can be split into multiple files.
- Logs and data can be piped into external applications.

nandump

Some older devices won't work with dd. You can download nandump from several web locations including github. You can then put it on the phone:

1. adb push nanddump /dev/ Examiner_Folder/ nanddump
2. chmod + x
3. /dev/ Examiner_Folder/ nanddump

nandump

You can then set the phone up to communicate:

adb forward tcp: 9999 tcp: 9999

Then from your workstation that is connected to the phone:
/dev/yourtargetfolder/nanddump

/dev/ block/ mmcblk0p34 | /dev/yourtargetfolder/
nc –l –p 9999

This technique is described in *Learning Android Forensics* by Rohit Tamma & Donnie Tindall

Where is data stored

Persistent data are stored to either the NAND flash, the SD card, or the network.

If stored internally the SQLite database is often used to store data. SQLite stores the entire database in a single file that is not platform (OS) dependent.

Preferences are often stored in an XML format.

Network storage is available to developers and requires the use of one of these packages:

- java.net.*
- android.net.*

Standard Partitions

- The boot loader partition is necessary for hardware initialization and loading the Android kernel. This is unlikely to have forensically important data.
- The boot partition has the information needed to bootup. Again, this is unlikely to have forensically important data.
- The recovery partition is used to boot the phone into a recovery console. While the partition may not have forensically relevant data, sometimes you may need to boot into recovery mode.

Standard Partitions

- The userdata partition is the one most relevant to forensic investigations. Here you will find the majority of user data including all the data for apps.
- The cache partition stores frequently accessed data and recovery logs. This can be very important for forensic investigations.
- The system partition is not usually important for forensic examinations.

Problems

Sometimes you get a dump of a phone and still cannot analyze it

One reason can be the out of band area. The OOB area is a small section of the flash memory reserved for metadata. The metadata usually includes an error correcting code (ECC), information about bad blocks, and sometimes information about the filesystem. If your analysis tool does not account for the OOB area it can cause problems analyziing

.

Backup your phone with adb

```
C:\projects\teaching\Android\tools>cd platform-tools

C:\projects\teaching\Android\tools\platform-tools>adb devices
List of devices attached
LGL16C9f11b9bf  device
```

```
C:\projects\teaching\Android\tools\platform-tools>adb backup -all -f c:\phonebackup.ab
Now unlock your device and confirm the backup operation...

C:\projects\teaching\Android\tools\platform-tools>
```


Nandroid backup

- The operating system itself (so you can make a copy of your stock or custom ROM and return to it if desired)
- All apps (including those you installed yourself or that came with the device)
- All games and your progress in them
- All pictures
- All music
- All videos
- All text and picture messages
- All wallpapers
- All widgets
- All ringtones
- All login and account settings
- All system settings
- All stored passwords, including Wi-Fi passwords

Nandroid backup

- With TWRP https://android.gadgethacks.com/how-to/twrp-101-make-nandroid-backup-restore-your-entire-phone-0175300/
- Download and install twrp on your phone
- This process will vary depending on your device, but for most phones, start by powering the device completely off. When the screen goes black, press and hold the volume down and power buttons simultaneously.
- You should see Android's bootloader
- use your volume buttons to highlight the "Recovery Mode" option, then press the power button to select it.
- Next, from TWRP's main menu, start by tapping the "Backup" button. After that, you see a list of check boxes—make sure that the "Boot," "System," and "Data" options are selected here. Finally, just swipe the slider at the bottom of the screen to start the backup process

What is Rooting?

- Administrative or root access
- Android by default doesn't have root
- Different Methods for different phones
- Rooting gives you a lot of advantages.

Root the phone

http://www.kingoapp.com/root-tutorials/how-to-root-android-without-computer.htm

- adb -d install KingoRoot
- adb -d install BusyBox.apk
- Then run both on the device.

Rooting

- First install busy box and root
- adb -d install BusyBox.apk
- then test:
- adb -d shell
- ls /data
- su
- ls /data

- We use ‘ls /data’ to test if we have access to a protected directory.

- The first time you run it, it should fail. Next we use ‘su’ to switch the user to root.
- We then use ‘ls /data’ again to test if we now have access to protected directories.

Flashing

- Installing something on your device
- Done through recovery or through ADB(Android Debug Bridge)
- It can be Rom/Kernel or recovery

Bootloader

- Lowest level of S/W on your phone
- Runs all the code to start OS
- Security Checkpoint for different partitions
- Locked Bootloader keeps phone safe
- Bootloader verify signature of system image before booting

Android Browser

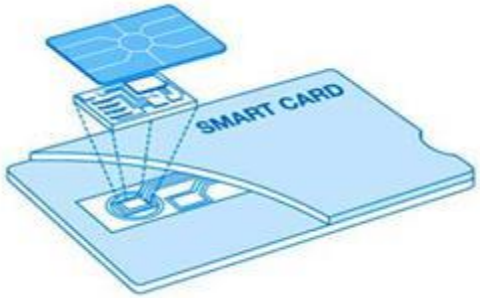
- Default browser is Google Chrome, but Chrome is based on the Blink browser engine. Blink is a fork of the WebCore component of WebKit.
- Blink has been supported since Android version 4.4
- Blink is also used as the engine for Opera, Vivaldi, Amazon Silk, and several others.

Authentication: Passwords

- Cheapest, easiest form of authentication
- Works well with most applications
- Also the weakest form of access control
  - Lazy users' passwords: *1234*, *password*, *letmein*, etc.
  - Can be defeated using dictionary, brute force attacks
- Requires administrative controls to be effective
  - Minimum length/complexity
  - Password aging
  - Limit failed attempts

Authentication: Smart Cards/ Security Tokens

- More expensive, harder to implement
- Vulnerability: prone to loss or theft
- Very strong when combined with another form of authentication, e.g., a password
- Does not work well in all applications

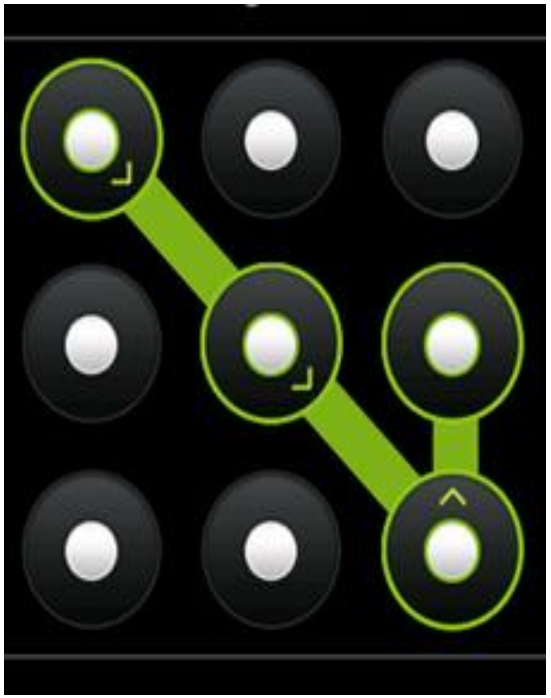


Authentication: Biometrics

- More expensive/harder to implement
- Prone to error:
  - False negatives: not authenticate authorized user
  - False positives: authenticate unauthorized user
- Strong authentication when it works
- Does not work well in all applications
  - Fingerprint readers becoming more common on mobile devices

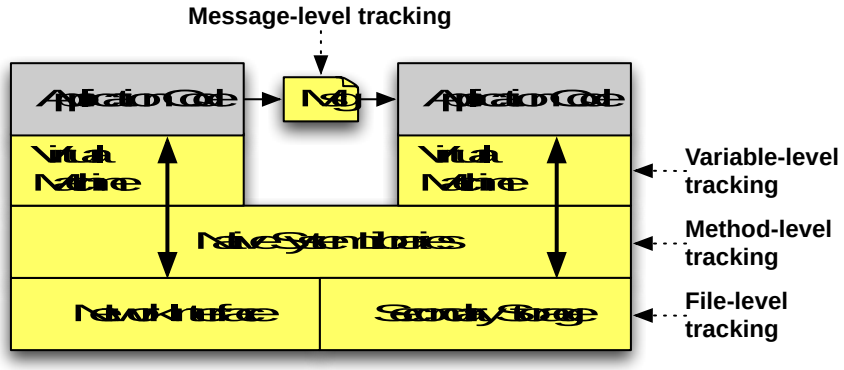
Authentication: Pattern Lock

- Swipe path of length 4–9 on 3 x 3 grid
- Easy to use, suitable for mobile devices
- Problems: [30]
  - 389,112 possible patterns; (456,976 possible patterns for 4-char case-insensitive alphabetic password!)
  - Attacker can see pattern from finger oils on screen



TaintDroid

- Information flow tracking https://www.usenix.org/legacy/event/osdi10/tech/full_papers/Enck.pdf
  - System firmware (*not* app)
  - Modifies Android's Dalvik VM, tracks info flows across methods, classes, files
  - Tracks the following info:
    - Sensors: GPS, camera, accelerometer, microphone
    - Internal info: contacts, phone #, IMEI, IMSI, Google acct
    - External info: network, SMS
  - Notifies user of info leakage



Source: [33]

Android Automotive

- Android Automotive OS is an Android-based infotainment system that is built into vehicles. The car's system is a stand-alone Android device that is optimized for vehicles. The operating system was first announced by Google in March 2017.
- https://source.android.com/devices/automotive

Android Automotive

- **Audi** Audi offers Android Auto in the Q5, SQ5, Q7, A3, A4, A5, A6, A7, R8, and TT
- **Acura** Acura offers Android Auto on the NSX
- **BMW** *BMW has announced that Android Auto will be available in the future, but has yet to release it*
- **Buick** Buick offers Android Auto in the Regal, LaCrosse, Envision and Encore
- **Cadillac** Cadillac offers Android Auto in the ATS, CTS, XTS, ELR, Escalade, XT5, and CT6
- **Chevrolet** Android Auto is available on the Chevrolet Spark, Sonic, Bolt, Camaro, Colorado, Corvette, Cruze, Equinox, Impala, Malibu, Silverado, Suburban, Tahoe, Trailblazer, Traverse, Trax, and Volt
- **Chrysler** Chrysler now offers Android Auto in the 300 and the Pacifica
- **Dodge** Dodge now offers Android Auto in the Charger, Challenger, and Durango
- **FIAT** Android Auto is available on the FIAT 500 and 500L
- **Ford** Ford offers Android Auto on the 2017 Fiesta, Focus, Fusion, Taurus, Mustang, C-Max, Escape, Edge, Explorer, Expedition, Flex, F-150, Transit Connect, and Super Duty
- **Genesis** Hyundai's new luxury brand offers Android Auto on its G80

Android Automotive

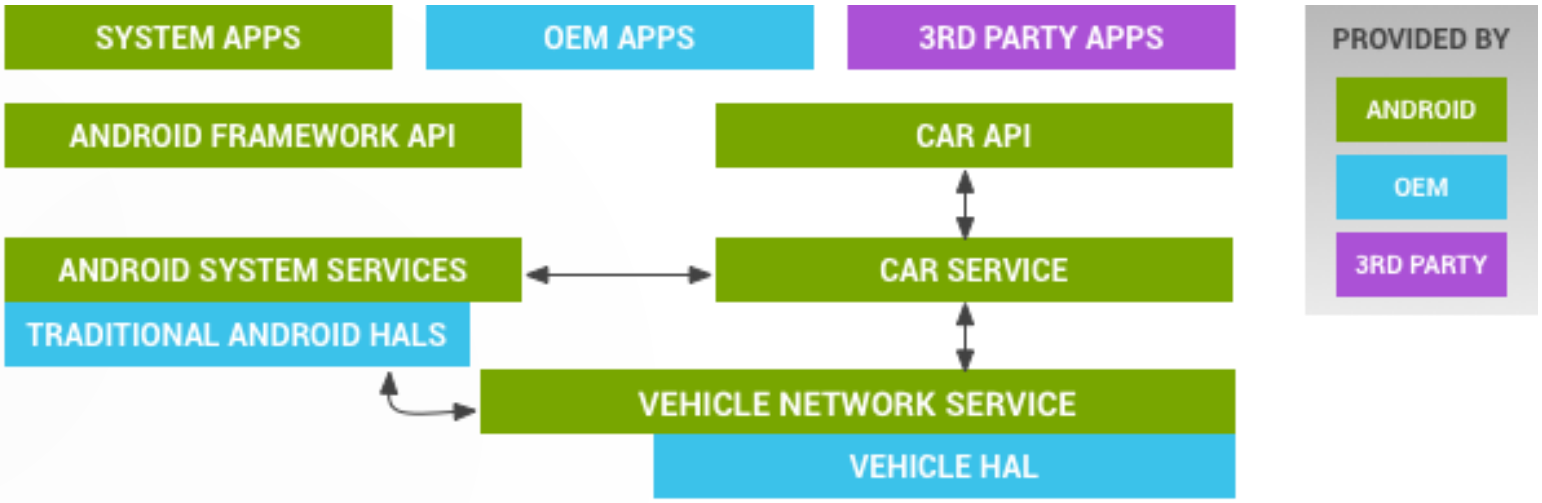
- **GMC** GMC offers Android Auto on the Canyon, Acadia, Sierra and Yukon models
- **Honda** Android Auto is available on the 2017 Accord, Ridgeline, Pilot, CR-V, Clarity Fuel Cell, and Civic.
- **Hyundai** Hyundai offers Android Auto on the Azera, Sonata, Tucson, Santa Fe, Santa Fe Sport, Elantra, and Ioniq
- **Infiniti** *Infiniti has not yet announced when Android Auto will be available.*
- **Jeep** Android Auto is offered on the Jeep Compass, Grand Cherokee, and Wrangler
- **Kia** Kia plans offers Android Auto in the Cadenza, Sorento, Forte, Forte5, Niro, Optima, Sedona, Soul, Sportage, Rio, and K900
- **Land Rover** *Land Rover has not yet announced if or when Android Auto will be available.*
- **Lexus** *Toyota has indicated that it will not offer either Apple CarPlay or Android Auto in any of its Toyota or Lexus vehicles for the foreseeable future.*
- **Lincoln** Lincoln offers Android Auto on its Continental, MKC, MKX, MKZ, and Navigator models.
- **Maserati** Maserati offers Android Auto on its Levante, Ghibli, and Quattroporte models.
- **Mazda** *Mazda has not yet announced when Android Auto will be available.*

Android Automotive

- **Mercedes-Benz** Mercedes offers Android Auto in the B-Class, C-Class. CLA-Class, CLS-Class, E-Class, GLA-Class, GLE-Class, GLS-Class, GLC Class, S-Class and SLC-Class.
- **MINI** *MINI has not yet announced when Android Auto will be available, but it likely will be available at some point.*
- **Mitsubishi** Mitsubishi offers Android Auto on the Mirage, Mirage G4, Outlander, and Outlander Sport models
- **Nissan** *Nissan has not yet announced when Android Auto will be available*
- **Porsche** *Porsche has not yet announced when Android Auto will be available*
- **RAM** Android Auto is available onthe RAM 1500*, 2500, 3500 and Chassis Cab*
- **Subaru** Subaru offers Android Auto on the BRZ, Impreza, and Legacy Outback
- **Toyota** *Toyota has indicated that it will not offer either Apple CarPlay or Android Auto in any of its Toyota or Lexus vehicles for the foreseeable future.*
- **Volkswagen** Volkswagen offers Android Auto on the Atlas, Beetle, CC, Golf, Golf GTI, Golf Sportwagen, Golf Alltrack, Golf R, e-Golf, Jetta, Passat, and Tiguan
- **Volvo** Volvo offers Android Auto in the XC60, XC90, S90, V90, and V90 Cross Country models.

Android Automotive

- Architecture



Android Automotive

hardware/libhardware/tests/vehicle/vehicle-hal-tool.c
Command-line native tool to load vehicle HAL and do simple operations. Useful for getting the system up and running in the early stages of development.

packages/services/Car/tests/carservice_test/
Contains car service testing with mocked vehicle HAL properties. For each property, expected behavior is implemented in the test. This can be a good starting point to understand expected behavior.

hardware/libhardware/modules/vehicle/
A basic reference implementation.

Android Auto - Audio

Automotive audio systems handle the following sounds and streams:

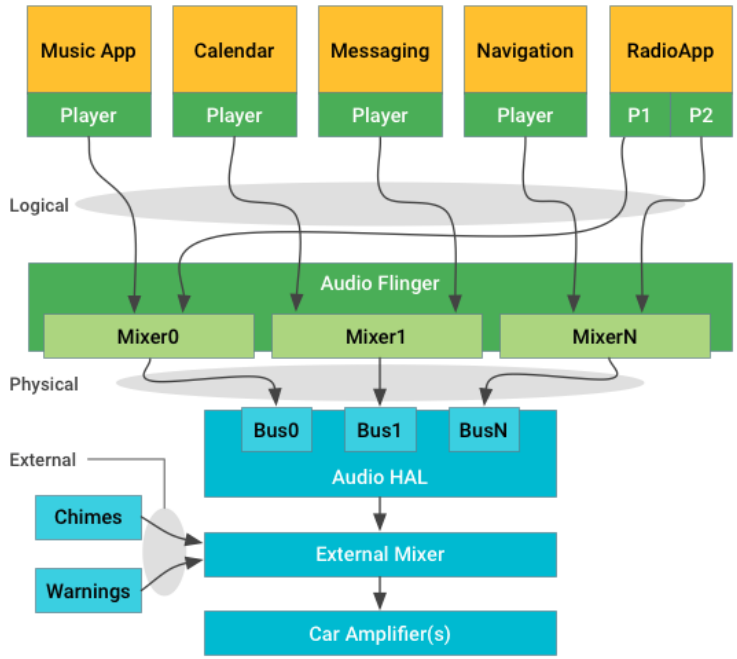


Figure 1. Stream-centric architecture diagram

Android is responsible for sounds coming from Android applications, controlling those applications and routing their sounds to individual streams at the HAL based on the type of sound:

- **Logical** streams, known as *sources* in core audio nomenclature, are tagged with Audio Attributes.
- **Physical** streams, known as *devices* in core audio nomenclature, have no context information after mixing.

Android TV

- First released in 2014 with Nexus Player. Used by Smart TV vendors including Sony and Sharp. https://www.android.com/tv/ Also used in set top boxes

Android TV

- Uses the TV Input Framework
- https://source.android.com/devices/tv It has several components:
  - TV Provider (com.android.providers.tv.TvProvider): a database of channels, programs, and associated permissions
  - TV App (com.android.tv.TvActivity): the app that handles user interaction
  - TV Input Manager (android.media.tv.TvInputManager): allows the TV Inputs to communicate with the TV App
  - TV Input: an app representing physical or virtual tuners and input ports
  - TV Input HAL (tv_input module): a hardware definition that allows system TV Inputs to access TV-specific hardware when implemented
  - Parental Control: the technology to allow blocking of channels and programs
  - HDMI-CEC: the technology to allow remote control of various devices over HDMI

Android TV

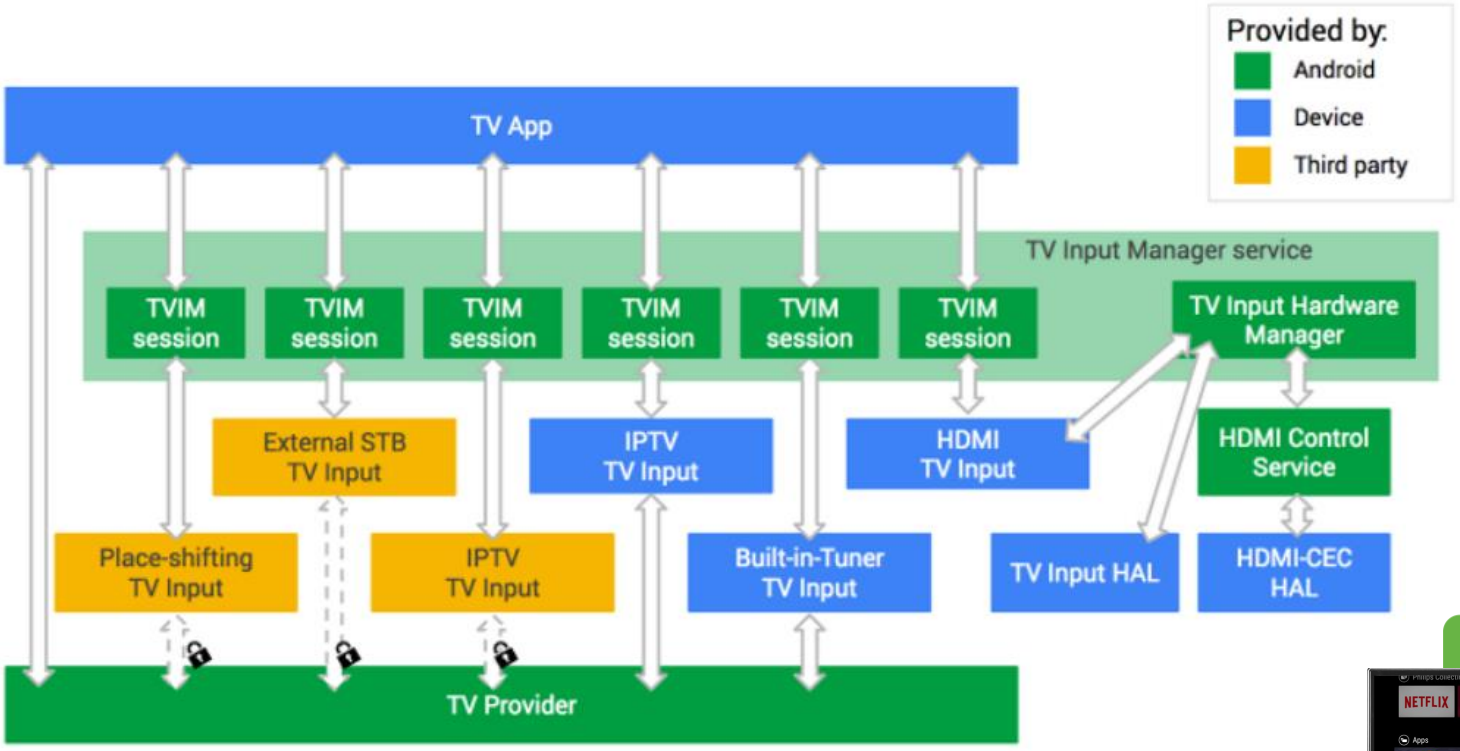


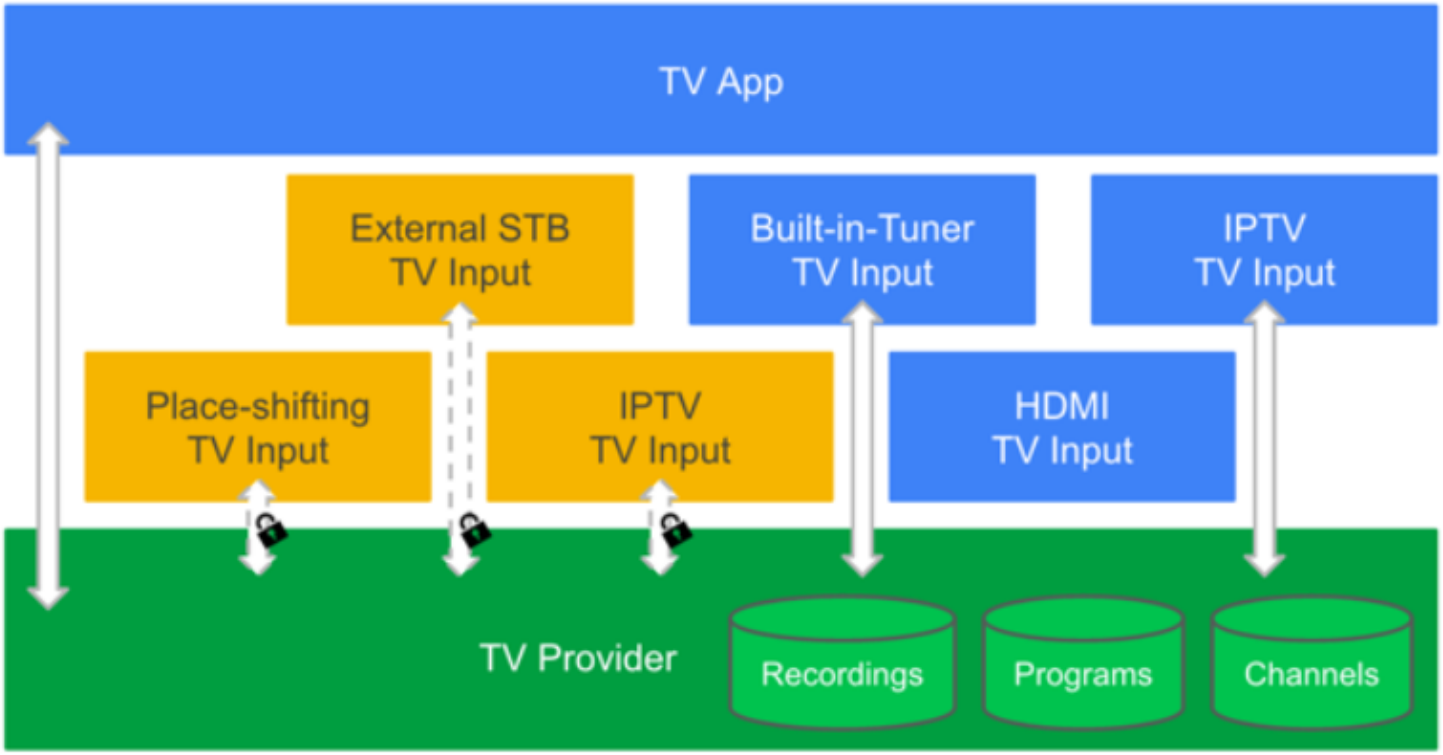
Figure 1. Android TV Input Framework (TIF) architecture

Android TV

The TV Provider database stores the channels and programs from TV Inputs. The TV Provider also publishes and manages the associated permissions so that TV Inputs can see only their own records. For instance, a specific TV Input can see only the channels and programs it has supplied and is prohibited from accessing any other TV Inputs' channels and programs.

The TV Provider maps "broadcast genre" to "canonical genre" internally. TV Inputs are responsible for populating "broadcast genre" with the value in the underlying broadcast standard, and the "canonical genre" field will automatically be populated with the correct associated genre from android.provider.TvContract.Genres. For example, with broadcast standard ATSC A/65 and program with genre 0x25 (meaning "Sports"), the TV Input will populate the "broadcast genre" with the String "Sports" and TV Provider will populate the "canonical genre" field with the mapped value android.provider.TvContract.Genres.SPORTS.
https://source.android.com/devices/tv

Android TV



Provided by:
- Android
- Device
- Third party

Figure 2. Android TV Provider