# Mobile Computing Technology

January 13, 2025

# Introduction

# Learning Objectives

1. Android / Linux

What is

# ANDROID

# Definition

- Linux is a fully open source operating system,
- Linux's kernel started as a Master's Thesis of Linus Torvalds
- Has since gained worldwide fame and adoption
- Available on many architectures including Standard IBM PC, ARM, MIPS and PowerPC architectures

# Android

- Built on top of Linux, but modifies it in substantial ways
- Provides a full software stack
  - a self-contained GUI environment
  - a rich set of frameworks – a prewritten, well-tested code to access advanced functionality - such as cameras, motion sensors, GUI widgets and more - in a few lines of code
  - Java ad Kotlin as simple to use development languages
- Google maintains the frameworks and runtime of the AOSP (Android Open Source Project)

# Android/Linux

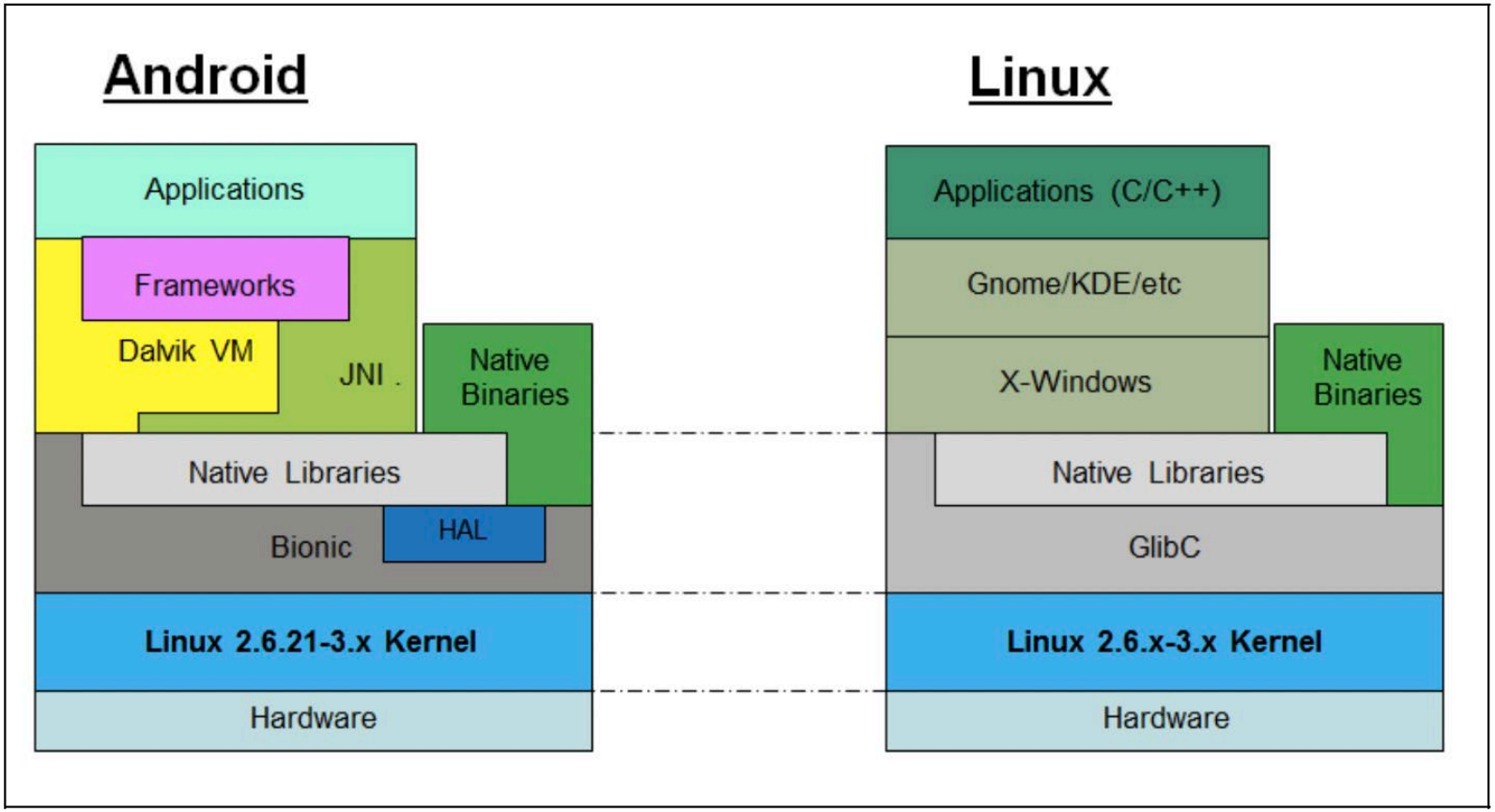
- A safe estimate would be that Android and Linux are about 95% alike at the kernel level, and about 65% or so at the user-mode
- What is a kernel?

# Android/Linux

- At the user-mode level, there is more of a divergence
  - introducing two entirely new components
    - the Dalvik runtime and
    - the Hardware Abstraction Layer
  - replacing glibc with ***Bionic***,
  - providing a custom version of the *init* -- the *system startup daemon*.
  - The underlying OS, however, still remains for the most part unmodified, with native binaries, processes and threads behaving as on Linux.

# Android/Linux

**Figure 1-2:** The Android Architecture, compared with that of mainstream Linux



Discuss the unknown

# TERMS

# Clarifying terms

- OS
- Kernel
- Unix
- Glibc
- Hardware Abstraction Layer
- The *init*
- Open Source Project
- GUI / Widgets
- Native binaries
- Processes and Threads
- ????

What are

# ANDROID FRAMEWORKS

# Android Frameworks

- By providing the frameworks, Android facilitates the application creation process,
  - allowing developers to use higher-level languages such as Java, rather than low-level C/C++.
- Developers can draw on the plentiful APIs which handle graphics, audio and hardware access.
- Unlike X-Windows and GNOME/KDE, Android's graphics programming API are far simpler

# Android Frameworks

- Packaged in different *namespaces,* according to their functionality.
- Packages in the *android.** namespace are available for use by developers.
- Packages in *com.android.** are internal.
- Android also supports most of the standard Java runtime packages in the *java.** namespace
- The entire set of frameworks is bundled into several Java ARchive (.jar) files on the device, in the directory names */system/framework*

# Android Frameworks

Chapter I: Introduction

**Table 1-2:** The Android Frameworks

| Package Name | API | Contents |
|---|---|---|
| android.app | 1 | Application Support |
| android.content | | Content providers |
| android.database | | Database support, mostly SQLite |
| android.graphics | | Graphics support |
| android.opengl | | OpenGL Graphics support |
| android.hardware | | Camera, input and sensor support |
| android.location | | Location support |
| android.media | | Media support |
| android.net | | Network support built over java.net APIs |
| android.os | | Core OS Service and IPC support |
| android.provider | | Built-in Android content-providers |
| android.sax | | SAX XML Parsers |
| android.telephony | | Core Telephony support |
| android.text | | Text rendering |
| android.view | | UI Components (similar to iOS's UIView) |
| android.webkit | | Webkit browser controls |
| android.widget | | Application widgets |
| android.speech | 3 | Speech recognition and Speech-to-Text |
| android.accounts | 4 | Support for account management and authentication. |

# Android Frameworks (cont.)

| | | |
|---|---|---|
| android.gesture | | Custom gesture support |
| android.accounts | 5 | User account support |
| android.bluetooth | | Bluetooth support |
| android.media.audiofx | 9 | Audio Effects support |
| android.net.sip | | Support for VoIP using the Session Initiation Protocol (RFC3261) |
| android.os.storage | | Support for Opaque Binary Blobs (OBB) |
| android.nfc | | Support for Near Field Communication |
| android.animation | 11 | Animation of views and objects |
| android.drm | | Digital Rights Management and copy protection |
| android.renderscript | | RenderScript (OpenCL like computation language) |
| android.hardware.usb | 12 | USB Peripheral support |
| android.mtp | | MTP/PTP support for connected cameras, etc |
| android.net.rtp | | Support for the Real-Time-Protocol (RFC3501) |
| android.media.effect | 14 | Image and Video Effects support |
| android.net.wifi.p2p | | Support for Wi-Fi Direct (Peer-To-Peer) |
| android.security | | Support for keychains and keystores |
| android.net.nsd | 16 | Neighbor-Service-Discovery through Multicast DNS (Bonjour) |
| android.hardware.input | | Input device listeners |

* - This table, while detailed, is not comprehensive, and only reflects the more important classes. A full list can be found at http://developer.android.com/sdk/api_diff/##/changes.html, replacing ## with the API level

# Android Frameworks (cont.)

**Table 1-2 (cont.):** The Android Frameworks

| Package Name | API | Contents |
|---|---|---|
| android.hardware.display | 17 | External and virtual display support |
| android.service.dreams | | "Dream" (screensaver) support |
| android.graphics.pdf | 19 | PDF Rendering |
| android.print[.pdf] | | Support for external printing |
| android.app.job | 21 | Job scheduler |
| android.bluetooth.le | | Bluetooth Low-Energy (LE) support |
| android.hardware.camera2 | | The new camera APIs |
| android.media.[browse/projection/session/tv] | | Media browsing and TV support |
| android.service.voice | | Activation by "hot words" (e.g. "OK Google") |
| android.system | | `uname()`, `poll(2)` and `fstat[vfs](2)` |
| android.service.carrier | 22 | SMS/MMS support (CarrierMessagingService) |

# Discussion

- ***namespaces***
- ***Java***
- Software ***packages*** in Java
- Standard ***Java runtime*** packages

Just Google the terms or check Wikipedia and answer (your best guess is just fine)

The

# DALVIK VM

# *Dalvik* VM and Android RunTime

- A notable addition to Linux is the introduction of the Dalvik Virtual Machine.
- Android Applications run in the virtual machine
- Similar to a *Java VM*
  - it runs a different form of bytecode (called DEX, for Dalvik *Executable*),
  - is more optimized for efficiency and sharing memory than the JVM designed by Sun/Oracle.
  - is based on a free open source clone of JVM.

# Android RunTime

- The main layer in Android - Dalvik - involves significant processing, and even its many enhancements (e.g JIT compilation) still require much more work than native code would.
- Rather than using a VM and JIT compilation, the Android RunTime (ART) uses Ahead-of-Time (AOT) compilation.

- Higher Performance

# Java Native Interface

- Allows the inclusion of native libraries in application code, through the Java Native Interface (JNI)
  - e.g., C code written to access the graphics chip on a particular mobile device
  - Non native code are written to run on the VM and are agnostic to the underlying architecture (e.g., Intel, ARM, MIPS), providing greater portability of apps across various mobile devices.
- Google provides the Native Development Kit (NDK) which developers can use to build native libraries

# Discussion

- ***Virtual Machine***
- ***Executable***

What are

# NATIVE BINARIES

# Native Binaries

- Android's critical system component are implemented in C/C++, and are compiled into *native binaries*.
- User applications are compiled into Dalvik bytecode,
  - the bytecode runs in the context of a Dalvik Virtual machine
- The VM itself is an ELF binary
- Binaries are usually located in */system/bin*,
  - and */system/xbin*
  - with a few critical binaries located in */sbin*.

# Discussion

- The file/directory hierarchy in Unix
- What is the root directory ‘/’?
- Open your VM (Lubunto)
- Open a terminal
- type
  - cd /
  - type ‘ll’ or ‘ls –a’ and then hit Enter

What is

# BIONIC

# Glibc and Bionic

- When you develop a C code and want to use some functions you typically #include <stdlib.h> and other headers
- When you build your code the *libc.so* is linked to your output file
- Linux distributions use GNU's LibC (GLibC) as their core runtime (the binary library file named libc.so),
- Android elects to use its own C-runtime library, which is called ***Bionic***.

# Glibc and Bionic

- ***Bionic***
  - simpler and more lightweight than GlibC
  - more efficient for Android's purposes, leaving out features deemed unnecessary or too complicated
    - Omits some system calls
    - No support for UNIX System V Interprocess Communication, including semaphores and shared memory. Android provides it's own version of shared memory and an **IPC** concept called *Binder*

# Glibc and Bionic (cont.)

- ***Bionic***
  - Pthread support is built-in to Bionic, but limited functionality
  - No support for Thread cancellation, i.e., pthread_cancel() is not supported
  - Mutex is not supported
  - C++ Exceptions are not supported. The C++ Standard Template Library not supported

# Glibc and Bionic (cont.)

- **System Properties:** Properties are a unique feature of Android, which allow both the system as well as applications to supply various configuration and operational parameters in a simple key/value store.
  - Device Name
  - Path variables
- Android relies heavily on this mechanism, which is supplied through a shared memory region, accessible and read-only to all processes on the system, but settable only through /init.

What is the Android

# HAL

# Hardware Abstraction Layer

- Android defines a Hardware Abstraction Layer (HAL) which aims to promote standardization by defining an adapter

- The Hardware Abstraction Layer defines what an abstract camera, GPS, sensor, and other components look like to Android

- Hardware vendors must conform to the interface Android expects.

The Android

# DEVICE STORAGE

# Device Storage in Android

- The device's storage is broken up into disjoint chunks, each of them individually formatted and purposed.
  - Chunks are called ‘partitions’
  - /system is where the OS itself is installed
  - /data is where user data is stored,
  - Linux pseudo-filesystems
    - not a part of Android; serve important functions during system operation, primarily for diagnostics and hardware access

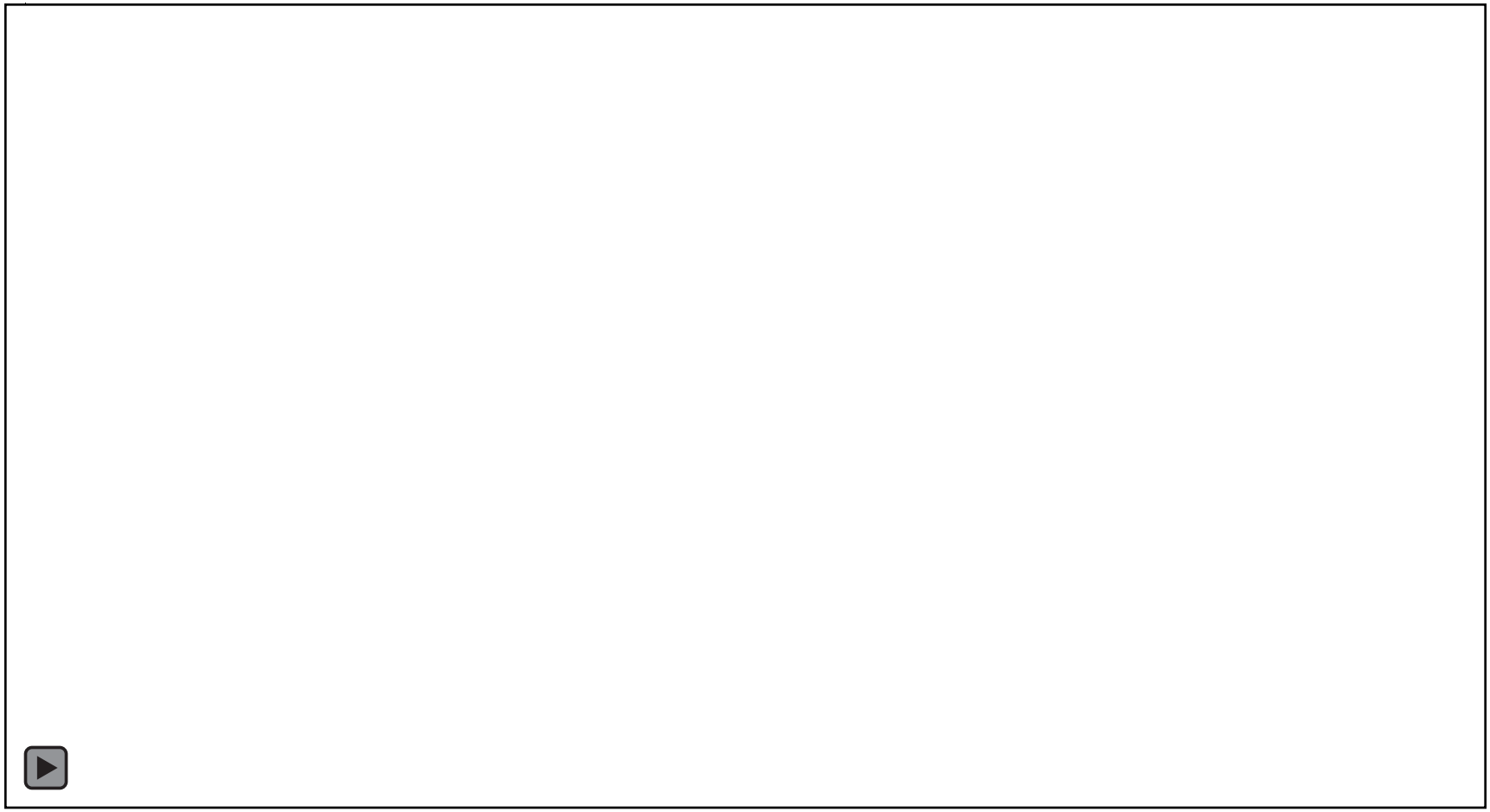
# Partitions

- ***cd /proc***
- the file ***/proc/partitions*** contains information about the available partitions (the the partition map)
- Alternatively, you can use the ***df*** command to explore the partitions that are mounted
- A partition must be mounted before it is visible / accessible

# Classroom Experiments

- Install Android Studio
- Pair your phone with Android – Check Android Manuals on How to Pair your phone with your computer
  - Follow classroom instructions; You need to make ‘developer options’ visible on your android phone. Then enable ‘USB debugging’; Then open Android Studio; Then pair the phone.
  - Then find where you have installed the Android ‘plattform-tools’
  - Then connect your phone with a USB cable to your computer
  - Then try the command-line instruction: ./adb shell
  - It will automatically give you a shell on the device that is paired with your computer

# Classroom Experiments (cont.)



# Classroom Experiments (cont.)

- login into your device shell
- type ***cat /proc/partitions***

```
[a53x:/ $ cat /proc/partitions
major minor  #blocks  name

   1        0       8192 ram0
   1        1       8192 ram1
   1        2       8192 ram2
   1        3       8192 ram3
   1        4       8192 ram4
   1        5       8192 ram5
   1        6       8192 ram6
   1        7       8192 ram7
   1        8       8192 ram8
   1        9       8192 ram9
   1       10       8192 ram10
   1       11       8192 ram11
   1       12       8192 ram12
   1       13       8192 ram13
   1       14       8192 ram14
   1       15       8192 ram15
   7        0      47172 loop0
   7        8        828 loop1
   7       16      40784 loop2
   7       24       8032 loop3
   7       32      37240 loop4
   7       40        268 loop5
   7       48        264 loop6
   7       56      37240 loop7
```

# Classroom Experiments (cont.)

**Output 2-4:** //dev/block/platform/.../by-name from a Nexus 5

```
shell@nexus5$ ls -l /dev/block/platform/msm_sdcc.1/by-name| cut -c56-
DDR -> /dev/block/mmcblk0p24
aboot -> /dev/block/mmcblk0p6       # Application Boot loader (Android Boot)
abootb -> /dev/block/mmcblk0p11     # Backup of Application Boot Loader
boot -> /dev/block/mmcblk0p19       # Kernel + InitRAMFS used to boot system
cache -> /dev/block/mmcblk0p27      # Mounted as /cache (used for updates/recovery)
crypto -> /dev/block/mmcblk0p26
fsc -> /dev/block/mmcblk0p22
fsg -> /dev/block/mmcblk0p21
grow -> /dev/block/mmcblk0p29       # usually empty
imgdata -> /dev/block/mmcblk0p17    # Boot loader graphic images (imgdata format)
laf -> /dev/block/mmcblk0p18        # LG Advanced Flash Daemon
metadata -> /dev/block/mmcblk0p14   # usually empty
misc -> /dev/block/mmcblk0p15       # Reserved for system->Boot Loader communication
modem -> /dev/block/mmcblk0p1       #
modemst1 -> /dev/block/mmcblk0p12   # Modem (Radio) boot
modemst2 -> /dev/block/mmcblk0p13   #
pad -> /dev/block/mmcblk0p7         # usually empty
persist -> /dev/block/mmcblk0p16    # Persistent settings for system components
recovery -> /dev/block/mmcblk0p20   # Alternate kernel + InitRAMFS for recovery
rpm -> /dev/block/mmcblk0p3         # Resource/Power Management loader
rpmb -> /dev/block/mmcblk0p10       # Backup of Resource/Power Management loader
sbl1 -> /dev/block/mmcblk0p2        # Secondary Boot Loader
sbl1b -> /dev/block/mmcblk0p8       # Backup of Secondary Boot Loader
sdi -> /dev/block/mmcblk0p5
ssd -> /dev/block/mmcblk0p23
system -> /dev/block/mmcblk0p25     # mounted as /system
tz -> /dev/block/mmcblk0p4          # ARM TrustZone
tzb -> /dev/block/mmcblk0p9         # Backup of ARM TrustZone
userdata -> /dev/block/mmcblk0p28   # mounted as /data
```

# Classroom Experiments (cont.)

```
a53x:/dev/block/platform/13500000.ufs/by-name $ ls -l /dev/block/platform/13500000.ufs/by-name/
total 0
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 blcmd -> /dev/block/sda13
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 boot -> /dev/block/sda14
lrwxrwxrwx 1 root root 15 2024-01-27 16:02 bota -> /dev/block/sda8
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 cache -> /dev/block/sda33
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 cp_debug -> /dev/block/sda18
lrwxrwxrwx 1 root root 15 2024-01-27 16:02 cpefs -> /dev/block/sdd1
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 debug -> /dev/block/sda28
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 dqmdbg -> /dev/block/sda23
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 dtbo -> /dev/block/sda12
lrwxrwxrwx 1 root root 15 2024-01-27 16:02 efs -> /dev/block/sda1
lrwxrwxrwx 1 root root 15 2024-01-27 16:02 epbl -> /dev/block/sda4
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 fmp_selftest -> /dev/block/sda27
lrwxrwxrwx 1 root root 15 2024-01-27 16:02 harx -> /dev/block/sda9
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 keyrefuge -> /dev/block/sda22
lrwxrwxrwx 1 root root 15 2024-01-27 16:02 keystorage -> /dev/block/sda7
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 metadata -> /dev/block/sda26
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 misc -> /dev/block/sda21
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 nad_refer -> /dev/block/sda19
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 omr -> /dev/block/sda34
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 optics -> /dev/block/sda32
lrwxrwxrwx 1 root root 15 2024-01-27 16:02 param -> /dev/block/sda5
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 persistent -> /dev/block/sda20
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 prism -> /dev/block/sda31
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 radio -> /dev/block/sda17
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 recovery -> /dev/block/sda16
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 reserved -> /dev/block/sda29
lrwxrwxrwx 1 root root 15 2024-01-27 16:02 sec_efs -> /dev/block/sda2
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 spu -> /dev/block/sda35
lrwxrwxrwx 1 root root 15 2024-01-27 16:02 steady -> /dev/block/sda3
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 super -> /dev/block/sda30
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 uh -> /dev/block/sda10
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 uhcfg -> /dev/block/sda11
lrwxrwxrwx 1 root root 15 2024-01-27 16:02 up_param -> /dev/block/sda6
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 userdata -> /dev/block/sda36
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 vbmeta -> /dev/block/sda24
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 vbmeta_system -> /dev/block/sda25
lrwxrwxrwx 1 root root 16 2024-01-27 16:02 vendor_boot -> /dev/block/sda15
a53x:/dev/block/platform/13500000.ufs/by-name $
```

# Classroom Experiments (cont.)

```
NAME
     df – display free disk space

SYNOPSIS
     df [-b | -h | -H | -k | -m | -g | -P] [-ailn] [-t] [-T type]
        [file | filesystem ...]

LEGACY SYNOPSIS
     df [-b | -h | -H | -k | -m | -P] [-ailn] [-t type] [-T type] [file |
     filesystem ...]

DESCRIPTION
     The df utility displays statistics about the amount of free disk space on
     the specified filesystem or on the filesystem of which file is a part.
     Values are displayed in 512-byte per block counts.  If neither a file or
     a filesystem operand is specified, statistics for all mounted filesystems
     are displayed (subject to the -t option below).

     The following options are available:

     -a      Show all mount points, including those that were mounted with the
             MNT_IGNORE flag.

     -b      Use (the default) 512-byte blocks.  This is only useful as a way
             to override an BLOCKSIZE specification from the environment.

     -g      Use 1073741824-byte (1-Gbyte) blocks rather than the default.
             Note that this overrides the BLOCKSIZE specification from the
             environment.

     -H      "Human-readable" output.  Use unit suffixes: Byte, Kilobyte,
             Megabyte, Gigabyte, Terabyte and Petabyte in order to reduce the
             number of digits to three or less using base 10 for sizes.

     -h      "Human-readable" output.  Use unit suffixes: Byte, Kilobyte,
             Megabyte, Gigabyte, Terabyte and Petabyte in order to reduce the
             number of digits to three or less using base 2 for sizes.
```

# Android Device Partitions

**Table 2-1:** Android Standard Partitions

| Name | format | Notes |
|---|---|---|
| boot | bootimg | Kernel + initramfs. Contains kernel and RAMdisk to boot by default. |
| cache | Ext4 | Android's /cache: used for updates and recovery. |
| recovery | bootimg | Boot-to-recovery: Kernel + alternate initramfs to start system recovery. |
| system | Ext4 | Android's /system partition - OS Binaries and frameworks. |
| userdata | Ext4/F2FS | Android's /data partition - User data and configuration. |

Partitions on Android devices have set names, but most of them are quick cryptic. To complicate matters, different device chipsets and vendors use different partitions, as well as different names for the same functional partitions.

# Android Device Partitions

try on Samsung
cat /system/etc/fstab.postinstall

The kernel maintains a list of all supported filesystems in /proc/filesystems

cd /dev/block/platform/13500000.ufs/by-name/

ls -l | cut -c43-

# Chipset-specific Partitions

**Table 2-2:** Partitions found on Qualcomm MSM devices

| Name | Format | Notes |
|---|---|---|
| aboot | bootldr | Application Processor Boot:This contains the Android Boot Loader. Note some devices may use custom boot loaders instead (e.g. HTC's HBoot). |
| modem | MSDOS | Contains various ELF binaries and data files to support device modem |
| modemst[1\|2] | proprietary | Non-Volatile data for modem |
| rpm | ELF 32-bit | Resource Power Management: This provides the first stage bootloader |
| sbl[123] | Proprietary | Secondary Boot Loader optionally split into up to three stages. |
| tz | ELF 32-bit | ARM TrustZone |

Chipset vendors often require dedicated partitions for their components

# Vendor-specific Partitions

**Table 2-3:** Vendor custom partitions

| Name | Vendor | Notes |
|---|---|---|
| hboot | HTC | HTC's proprietary boot loader. Replaces aboot on HTC devices |
| efs | Samsung | Encrypted File System. Contains various configuration files |
| ssd | Samsung | Secure Software Download |
| ota,fota | Samsung | Firmware-Over-The-Air: Used in the process of phone updates |
| grow | Samsung, LG | Empty partition, to allow partition growth |
| laf | LG (G2, Nexus5) | Contains an alternate bootimg which loads lafd (LG Advanced Flash Daemon) used for device re-flashing. The laf partition is a recovery image format. |
| imgdata | LG (G-PAD, G2, Nexus5) | RLE images in IMGDATA format, similar to BOOTLDR |

blcmd -> /dev/block/sda13
**boot -> /dev/block/sda14**
bota -> /dev/block/sda8
**cache -> /dev/block/sda33**
cp_debug -> /dev/block/sda18
cpefs -> /dev/block/sdd1
debug -> /dev/block/sda28
dqmdbg -> /dev/block/sda23
dtbo -> /dev/block/sda12
**efs -> /dev/block/sda1**
epbl -> /dev/block/sda4
fmp_selftest -> /dev/block/sda27
harx -> /dev/block/sda9
keyrefuge -> /dev/block/sda22
keystorage -> /dev/block/sda7
metadata -> /dev/block/sda26
misc -> /dev/block/sda21
nad_refer -> /dev/block/sda19
omr -> /dev/block/sda34
optics -> /dev/block/sda32
param -> /dev/block/sda5
persistent -> /dev/block/sda20
prism -> /dev/block/sda31
radio -> /dev/block/sda17
**recovery -> /dev/block/sda16**
reserved -> /dev/block/sda29
sec_efs -> /dev/block/sda2
spu -> /dev/block/sda35
steady -> /dev/block/sda3
super -> /dev/block/sda30
uh -> /dev/block/sda10
uhcfg -> /dev/block/sda11
up_param -> /dev/block/sda6
**userdata -> /dev/block/sda36**
vbmeta -> /dev/block/sda24
vbmeta_system -> /dev/block/sda25
vendor_boot -> /dev/block/sda15

a53x:/dev/block/platform/13500000.ufs/by-name $ ls -l | cut -c43-

# Classroom Experiment

**Output 2-5:** Demonstrating `df` on a Nexus 9

```
shell@flounder:/ $ df
Filesystem               Size             Free   Blksize
/dev                   918.0M    32.0K   917.9M
/sys/fs/cgroup         91
/mnt/asec              918.0M     0.0K   918.
/mnt/obb               91
/system                  2.5G     1.6G   875.0M
/vendor                245.9M   149.3M    96.6M
/cache                 248.0M   256.0K   247.7M
/data                   11.0G     1.5G
/mnt/shell/emulated               1.5G
/storage/emulated      91
/storage/emulated/0     11.0G     1
/storage/emulated/0/Android/obb                      9.6G   4096
/storage/emulated/legacy    11.0G
/storage/emulated/legacy/Android/obb                      9.6G   4096
```

The "disk free" command shows total/used/available disk space for
each filesystem listed on the command line, or all currently mounted
filesystems.

The Android

# FILE SYSTEM

# File System

- The file systems are exposed to us in form of a ‘tree’
- The root is named ‘/’
- It contains files and directories
- Almost everything is a file
  - serial ports
  - network ports
  - shared memory
  - monitor
  - keyboard
  - sensors
  - ....

# File System

- Directories can contain files or other directories
- Root is a directory
- You can switch to the root directory using the following command
  - cd /
- Devices, processes, drivers, system resources etc are made available in special directories

# File System

- The kernel maintains a list of all supported filesystems in /proc/filesystems
  - lists which filesystems are supported either natively (i.e. compiled into the kernel) or as a loaded module
- Android's root file system is mounted from a RAM Disk (the "initramfs"). Upon every boot, the boot loaderder loads the filesystem image from the boot partition onto RAM, and provides it for the kernel.

# File System

| Directory | Notes | |
|---|---|---|
| /init | The binary launched by the kernel on startup as PID 1 | |
| fstab.hardware | The filesystem mount table used by fs_mgr | |
| init[...].rc | The configuration file(s) for /init. The main configuration file is always /init.rc, with optional additional files which are device and vendor dependent | |
| sbin/ | Contains critical binaries, such as adbd, healthd and (most importantly) recovery, which the system needs even if /system cannot be mounted. May also contain vendor binaries. | |

# File System

| Directory | Notes | |
|---|---|---|
| /init | The binary launched by the kernel on startup as PID 1 | |
| fstab.hardware | The filesystem mount table used by fs_mgr | |
| init[...].rc | The configuration file(s) for /init. The main configuration file is always /init.rc, with optional additional files which are device and vendor dependent | |
| sbin/ | Contains critical binaries, such as adbd, healthd and (most importantly) recovery, which the system needs even if /system cannot be mounted. May also contain vendor binaries. | |

# File System

**Table 2-7:** The contents of the /system partition

| Directory | Notes |
|---|---|
| app | System applications: These include the prebundled apps from Google, as well as any vendor or carrier-installed apps (though these should technically reside in /vendor/app, instead). |
| bin | Binaries: These include the various daemons, as well as shell commands (mostly links to toolbox, or - as of M - toybox). |
| build.prop | Properties generated as part of the build process. This file is sourced by init to load properties on boot |
| etc | Miscellaneous configuration files. Symlinked from /etc. q.v. Table fs-etc for contents. |
| fonts | True-Type Font (.ttf) files |
| framework | The Android frameworks. Frameworks are contained in .jar files, with their executable dex files optimized alongside them in .odex. |
| lib | Runtime libraries - native ELF shared object (.so) files. This directory serves the same role as /lib in vanilla Linux. |
| lost+found | Automatically generated directory for fsck operations on /system. Empty (unless the filesystem crashed, in which case it may contain unlinked inodes) |
| media | Alarm, notification, ringtone and UI-effect audio files in .ogg format, and the system boot animation (discussed in Chapter 5). |
| priv-app | Privileged Applications |
| usr | Support files, such as unicode mappings (icudt511.dat), key layout files for keyboards and devices, etc. |
| vendor | Vendor specific files, if any. Usually placed into subdirectories mirroring /system itself (e.g. bin/, lib/, and media/). |
| xbin | Special purpose binaries, not needed for normal operation (unlike those in bin. On the emulator, this is populated with various tools from the AOSPs /system/extras. On devices, this directory is normally empty, or contains only dexdump. Various rooting utilities drop "su" there as well. |

# Summary

- Linux/Android
- Clarification of technical terms
- Android Frameworks
- Native Binaries
- Bionic