An Introduction to Android OS

Lecture 2

Goal of this lecture

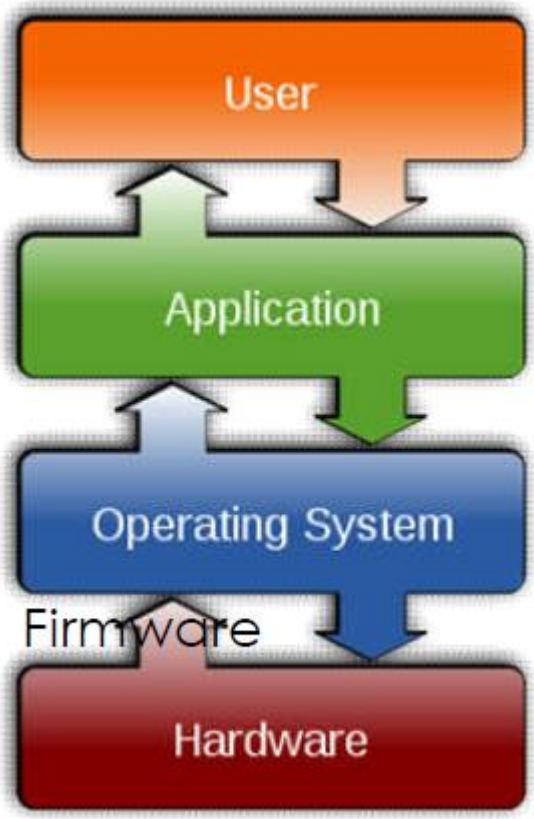
- Understand the architecture and constrains of a mobile operating system
- Become familiar with Android OS

Content

- Introduction to mobile operating systems
- Timeline of popular systems
- The contenders
- Why Android OS?
- Android specific components

Mobile OS

- A distinct class of embedded operating systems
- Optimized for code execution on mobile SoCs (with ARM CPU e.g., Bionic, Snapdragon, Kirin, Exynos)
- Interface for apps to access services provided by underlying hardware
- RTOS executed on a mobile system
- Mobile OS is pre-installed on the device



Mobile OS

Characteristics

- High security
- Stability and reliability
- Reduced power consumption
- Connectivity
- Interoperability
- Multitasking
- Flexibility in user interface (UI)
- Easy to use

Timeline of popular OS

- Palm OS (1996)
- Symbian (1998)
- Microsoft Pocket PC (2000)
- Windows Mobile (2003)
- Blackberry OS (2005)
- iOS (2007)
- iOS2, Android, Windows 6, Symbian 2 (2008)
- iOS3, Android 1.1, Blackberry 5, Web OS, Bada (2009)
- iOS4, Android 2.2, WPhone7, Symbian 3, MeeGo (2010)

Timeline of popular OS (continued)

- Tizen (Intel, Samsung, Linux) (2011)
- iOS6, Firefox (2012)
- iOS7, Android 4.4, Blackberry 10 (2013)
- iOS8, Android 5, WP 8.1 (2014)
- iOS9, Android 6, WP 10 (2015)
- iOS10, Android 7, Tizen 3 (2016)
- iOS11 (iPhone X), Android 8, Tizen 4 (2017)
- …. Android 8.1 & 9, 10, 11
- iOS16, Android 12; Win10 "Spring, Fall updates", Win11
  - New API features in Android 13.

Market share - total

- Google (Android) ➢74.25%
- Apple (iOS) ➢25.15%
- Samsung ➢0.23%
- Microsoft (Windows Phone) ➢0.03%
- Symbian ➢0.02%
- Bada ➢0.01%
- Tizen, Kindle, Blackberry ➢~0%

NetMarketShare
August 2020

Market share - trending

Recent surveys

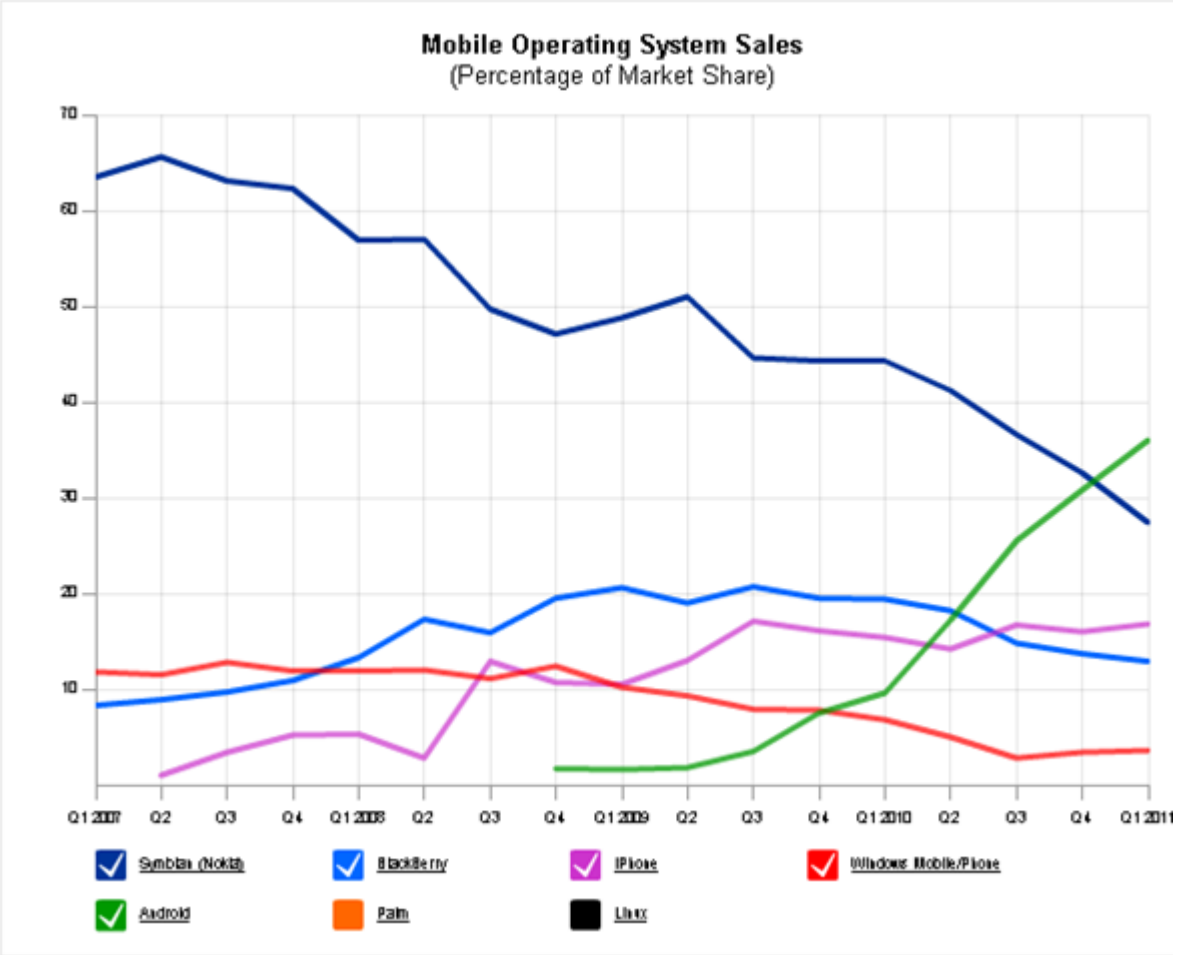
- Android+iOS = 96.8% of new phones (2015)
- Android+iOS = 99.6% of new phones (2016)

Some 2022 statistics

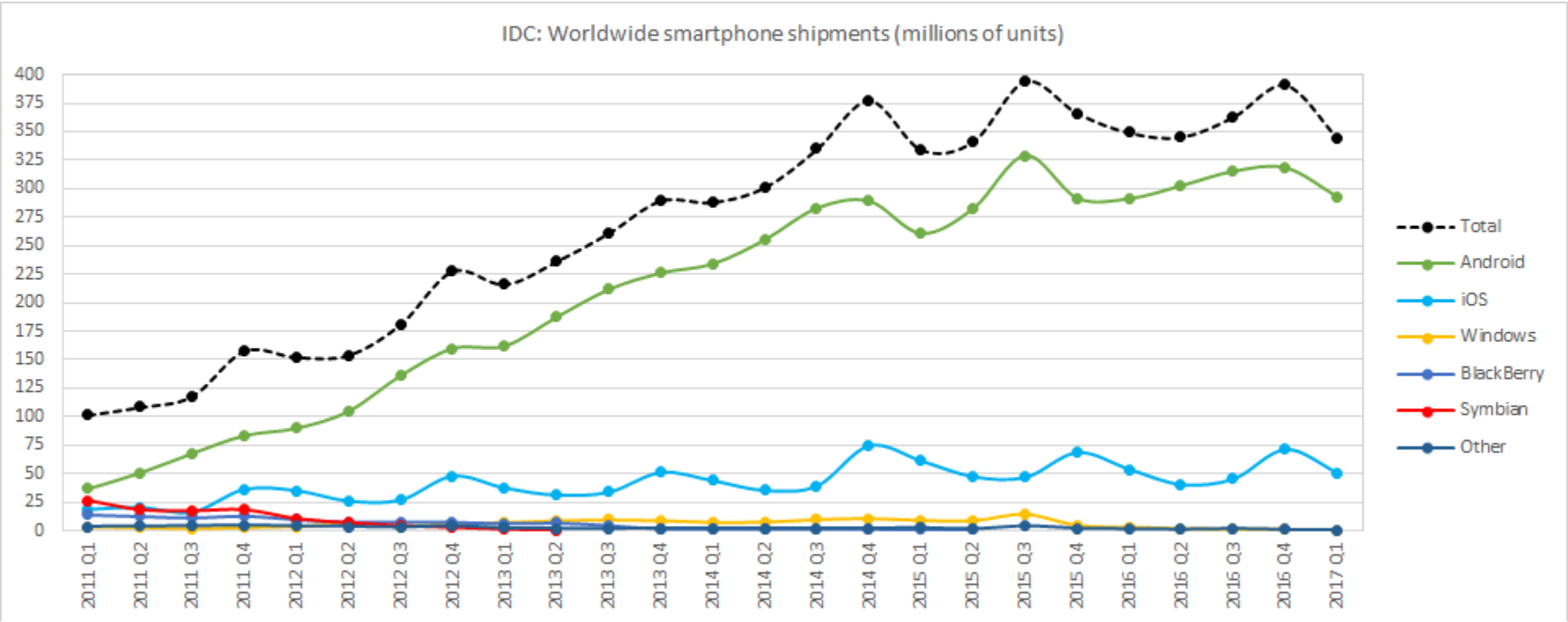
- Android sold 1B+ phones (86.2%)
- iOS sold 220M phones (13.8%)

The Verge
February, 2017

Popularity (2002-2011)



Popularity (2011-2017)



The Big Two

iPhone OS

- Closed platform (owned by Apple)
- Based on OS X, UNIX, with some open-source components
- Multitasking (since Phone 4)
- Programming languages – Objective C, Swift
- Third party applications can not replicate iPhone functionality
- Limited connectivity/synchronization
- Application installing via Apple store (a bit safer)
- Publishing process more difficult

The Big Two

Android OS

- Open platform
- Open source code
- Multitasking
- Development tools are free (Android Studio, Eclipse, Emulator)
- Java/Kotlin programming languages
- Applications installed via Google Play or directly (apk), but a bit more risky.

Android vs. iOS

iOS + : more secure & stable, 5yrs+ updates, App Store has highest quality apps, no bloatware

iOS - : OSX integration, rigid non-jailbroken, no variety in models, limited hardware, expensive (~~$400+~~ $1000)

Android + : customize everything, Google tech, more flexible, hardware variation (curved, headphone jack, keyboard), app variety

Android - : Google dependency, bloatware, less stable, OS fragmentation

Why Android OS?

…why Linux?

- Multiple variants, same source code
- Source code is free
- Robust and reliable
- Modular, configurable, scalable
- No licensing
- Large number of experimented developers
- Portability

The Android Operating System

- It is an software platform offered by Google and the Open Handset Alliance
- An open platform
- Based on Linux operating system
- Has an improved user interface, phone functions, multimedia support, user applications, etc.
- User applications are developed in Java (Kotlin):
  - Fast development
  - Robustness
  - Security

The Android Operating System

- Offers a complete software stack:
  1. Operating system
  2. Middleware
  3. User applications
- Dedicated and optimized for applications on mobile devices
- Application separation

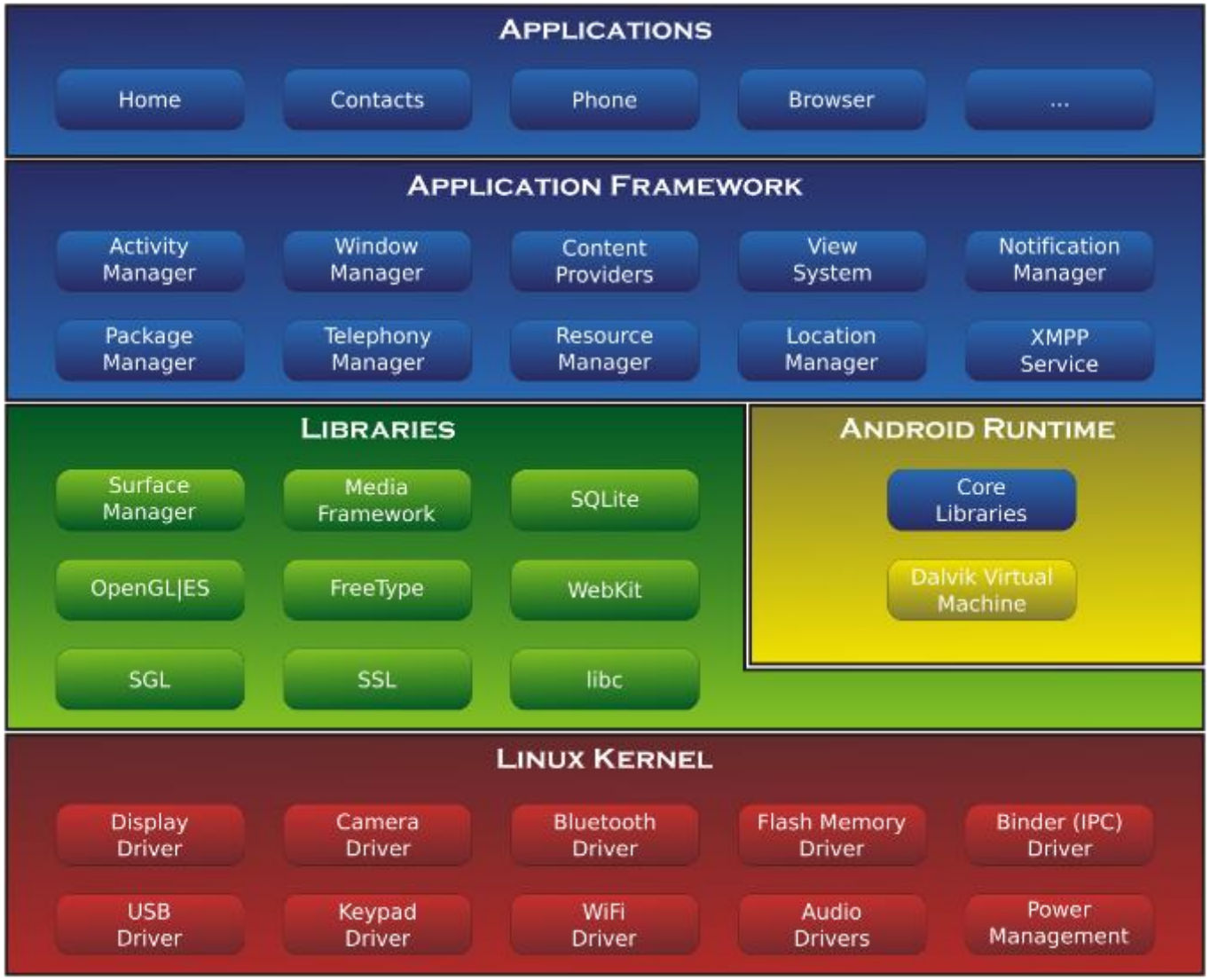
  Every application runs in its own VM (virtual machine), on their own user/group

Android OS architecture

System architecture is structured on several layers:

1. Linux kernel – system's core
2. Native libraries – libraries used by the different components
3. Android runtime - virtual machines in which user applications are executed
4. Application framework - mandatory services for the mobile device
5. User applications

Android OS architecture



Android OS Kernel

Based on Linux kernel (currently 5.4+)

- Does not include all functionalities and components of Linux core: windows system, glibc, etc.
- Serves like a hardware abstracting layer for higher software layers.

Android OS Kernel

Standard Linux Kernel provides:

- Memory management
- Process management
- Security model based on permissions
- File system and network I/O
- Network stack
- Device drivers (display, camera, Bluetooth, USB, wifi etc.)

Android OS Kernel

On top of that, Android kernel ensures:

- Mobile device memory management
- Support for shared libraries
- Power consumption management
- Debugger and logger
- Inter-process communication (*binder*)

Running Linux desktop apps on Android

Not possible because Android does not have a graphical server (X11) and not all GNU libraries are implemented (no shell).

To obtain a shell on Android, the device has to be *rooted* (e.g. using BusyBox).

Linux does not include the Dalvik VM.

- Ubuntu for Android tried to port Dalvik for desktop (failed)
- BlueStacks tries to do it for Windows and Mac.
- Need for an emulator

Android OS Libraries

C/C++ libraries are used by the different Android components

Developers can access them through application framework:

- Media Libraries: MPEG4, H.264, MP3, JPG, PNG
- WebKit/LibWebCore: web browser engine
- SQLite: relational database engine
- Libraries/engines for 2D and 3D graphics

Android OS Libraries

Summary:

- Surface manager – display on screen
- Media framework for playing audio/video files
- Webkit for displaying web pages
- OpenGL for higher performance graphics
- SQLite for managing relational DB in memory

Android OS Runtime

- Libraries which offer basic Java features.
- Dalvik Java virtual machine is based on the Linux core.
- Devices can run multiple Dalvik VMs.
- Every Android application is executed in its own Dalvik VM instance.
- The VM executes optimized executable files (.dex).
- Dx-tool transforms compiled Java files in dex-files.
- Offers applications a high grade of portability and execution consistency.

Dalvik VM

- Dalvik is the software executing the actual Android app code.

  Developer (Java code) → Java compiler (Java bytecode files) → DX (one single DEX file, classes.dex) → package manager (classes.dex+app resources) → Dalvik executes dex file

- Dalvik VM is used instead of JVM because it was designed and optimized for mobile devices (resource limitations)
- Dalvik is based on JIT (just in time) compilation

  Each time an app is running, the part of the code required for its execution is going to be translated (compiled) to machine code at the execution time

Dalvik VM

ART (Android RunTime) – experimental starting with Android 4.4, now used along Dalvik.

- Aims to boost the performance of Android apps
- Compiles the intermediate language, Dalvik bytecode, into a system-dependent binary
- The whole code of the app will be pre-compiled during install (once)
- AOT – ahead of time compilation, uses dex2oat tool

ART uses smarter garbage collection (GC):

- Parallel execution during collection
- Optimized for short-lived instances
- Reduced memory footprint and fragmentation
- Less frequent GCs

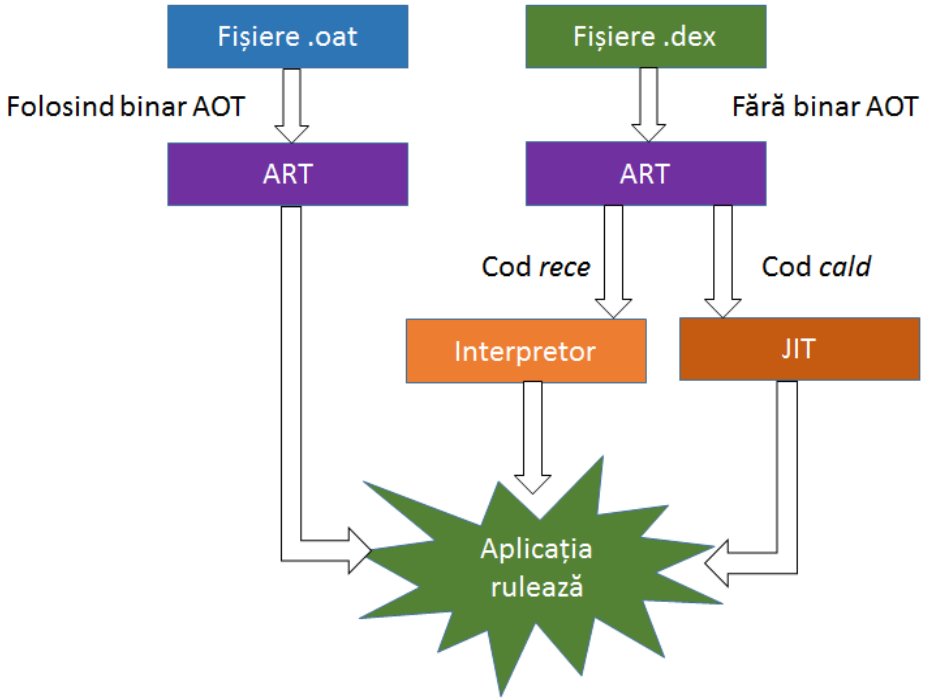
Dalvik VM

Android 7.0 adds JIT to ART's AOT!

JIT reduces memory storage space accelerates app refresh times

Compilation:

1. User starts app
2. ART is launched to load dex
3. If oat exists, it is executed
4. Else, ART executes dex through JIT or other interpreter



Application separation

Each app has its:

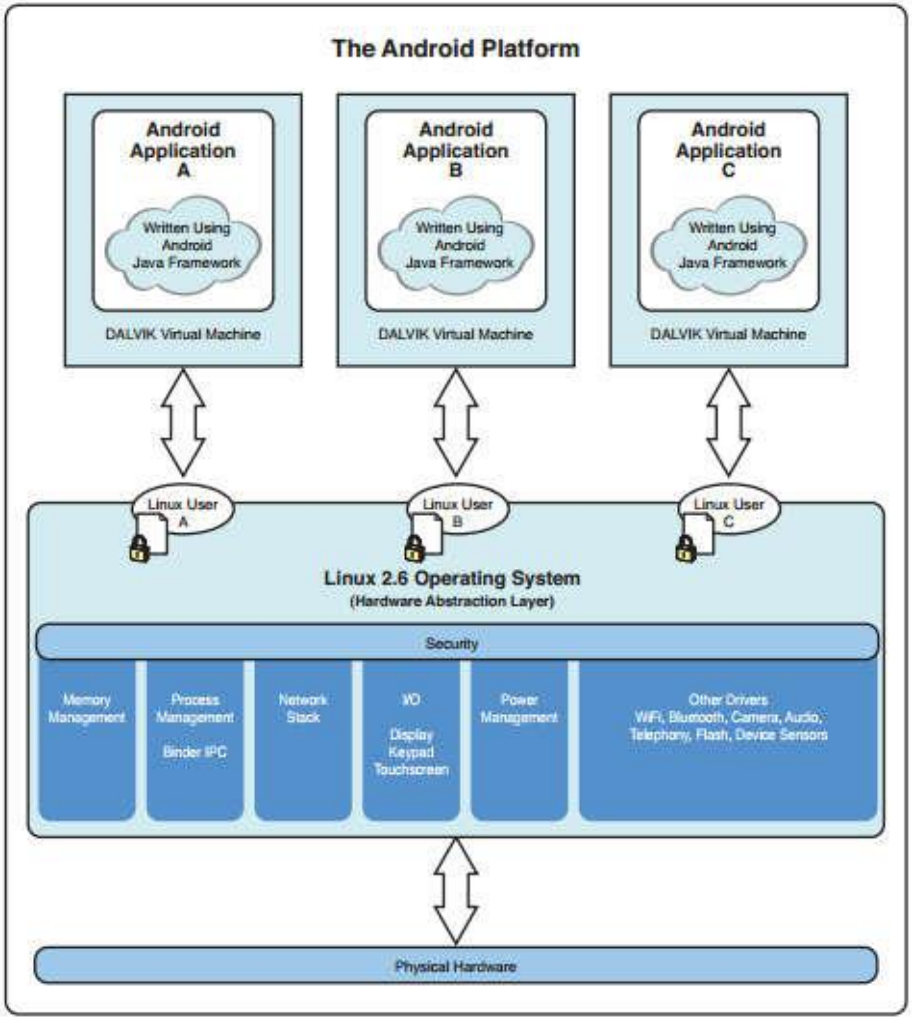
A. Own VM (sandboxed)

B. Own user (credential separation)

C. Separate process

D. Own rights

Advantages: security area (not sharing memory across multiple apps), failure isolation (a leak or crash will not *spread*)



Android OS Application framework

Mandatory services of a mobile device

- API for accessing system services
- Hardware services – allow access to low level API

Android OS Application framework

Contains hardware services like:

- Telephony Service
- Location Service
- Bluetooth Service
- WiFi Service
- USB Service
- Sensor Service

Android OS Application framework

And higher-level services like:

- Activity manager – manages the life cycle of an application
- Package manager – keeps information about installed applications in the system
- Window manager – manages all windows of an application
- View system – offers visual components mandatory for an application
- Content provider – database for sharing structured data between apps (e.g. contacts shared by dialer, sms, email, chat)
- Location manager – obtain position based on GPS, cell, wifi.
- Notification manager – placing messages in notification bar

Android OS Applications layer

- Predefined applications (implicit)
  Home screen launcher, dialer, browser, email client etc.
- User applications

User applications can replace existing applications

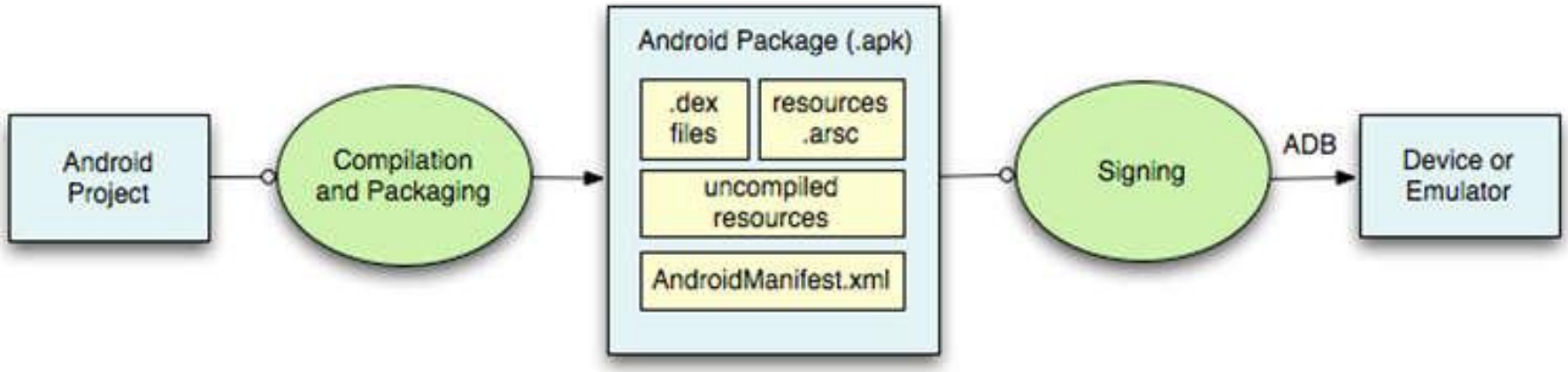
App = one single APK file, composed out of 3 components:

1. Dalvik executable (java code)
2. Resources (all non-code; xml, images, audio)
3. Native libraries (optional C/C++ runtime libraries)

Application signing

The process of creating an Android application:

- Code editing
- Application compilation and packaging
- Application signing
- Application delivery and installation



Application signing

Only signed APKs may be installed on an Android device.

For development ease, there is a debug key available.

When deploying (market release), a release key is needed.

Unlike other platforms (iOS, Windows), Android has distributed markets, with their own rules and policies

Google Play is the largest market, but there are also:

Amazon, Huawei, SlideME, 1Mobile market, Galaxy Apps, Opera Mobile store, Mobango, Soc-io Mall, F-droid, GetJar

What about malware?

With a distributed market, the chances rise to get infected with viruses and spyware.

Ex. Phishing cases reported through false banking apps

Android lets markets deal with their own problems =>

eventually, there will be markets with better reputation than others.

What is the motivation of Google?

- To help spread Android at every level of the market and business.
- Creating fair competition conditions for the mobile market.

Google is a media company which sells publicity!

→ Income not based on licensing, but on mediating information

→More Android users = more services sold through the platform

Est. $38BN revenue in 2011, $250BN in 2021…

Google's business model.