# Android Operating System:
## An in depth introduction

CS423 Project
Mohammad Alian, Shuomeng Guang, Bo Teng

# Outline

1. What is Android
2. History
3. Android architecture
4. Android vs Linux
5. Process Management
6. Power Management

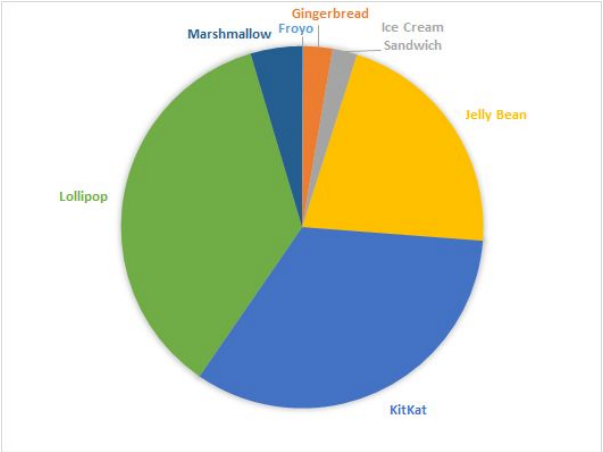
# What is Android OS?

- A mobile operating system developed by Google
- Based on Linux
- Designed Primarily for Mobile Apps like smartphones and tablets
  - Little memory
  - Slow processor
- Open Source Project @ https://source.android.com

Android OS is an operating system that was developed by Google for use on mobile devices. This means that it was designed for systems with little memory and a processor that isn't as fast as desktop processors. While keeping the limitations in mind, Google's vision for Android is that it would have a robust set of programming APIs and a very responsive UI. In order to facilitate this vision, they created an abstraction layer, which allows application developers to be hardware agnostic in their design.

# History

- Developed by Android, Inc (bought by Google in 2005)
- Unveiled in 2007
- Android 1. 0, 2008
- Developed by Open Handset Alliance
- Android 6.0 Marshmallow, 2015




| Codename | API | Distribution |
|---|---|---|
| Froyo | 8 | 0.10% |
| Gingerbread | 10 | 2.60% |
| Ice Cream Sandwich | 15 | 2.20% |
| Jelly Bean | 16-18 | 21.30% |
| KitKat | 19 | 33.40% |
| Lollipop | 21-22 | 35.80% |
| Marshmallow | 23 | 4.60% |

Android is the flagship software of the Open Handset Alliance (OHA), established by Google with members who are chip manufacturers, mobile handset makers, application developers or mobile carriers.
Clockwise development

Android was unveiled in 2007, along with the founding of the Open Handset Alliance – a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices.

# Android architecture

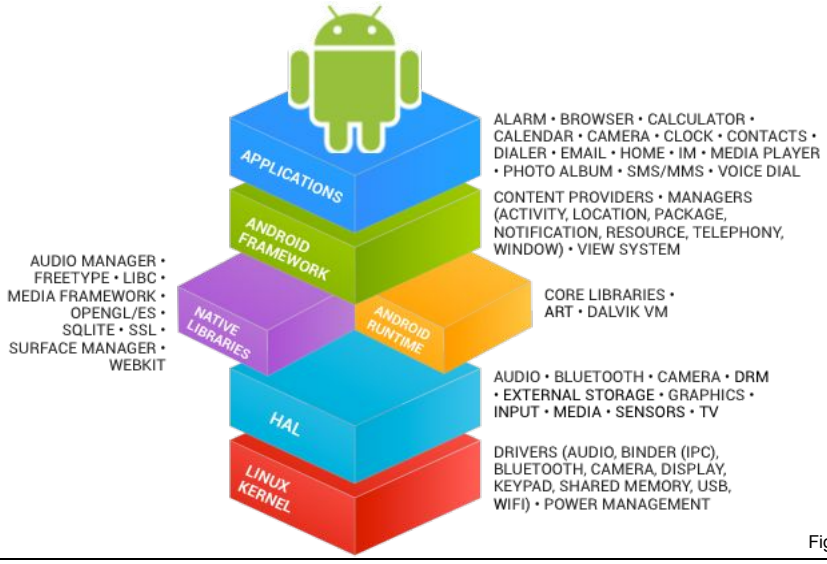



Figure from https://source.android.com/index.html

Binder IPC
The Binder Inter-Process Communication (IPC) mechanism allows the application framework to cross process boundaries and call into the Android system services code. This enables high level framework APIs to interact with Android system services. At the application framework level, this communication is hidden from the developer and things appear to "just work."

Android layers its PM model on top of standard Linux PM

## Linux Kernel

- Based on Linux Kernel
- Interact with hardware, contains hardware drivers
  - camera, audio, keypad, display...
- Process management
- Memory management
- Networking

  ...

- Additional specifications for mobile embedded platform (wake locks, the Binder IPC driver, power management)

the kernel provides preemptive multitasking, low-level core system services such as memory, process and power management in addition to providing a network stack and device drivers for hardware such as the device display, Wi-Fi and audio

Samsung galaxy 4 2015-11-01 5.0.1 kernel version 3.4.0

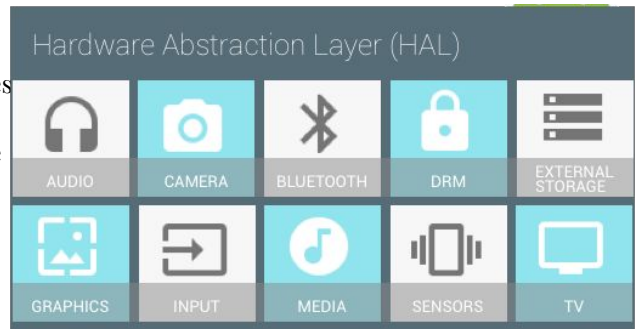
wake locks (a memory management system that is more aggressive in preserving memory)

Background on wakelocks
●Suspend mode powers down all hardware except memory
●One approach for optimized power management would be to suspend whenever idle
●Wakelocks are the primitive used to determine if the system is in use

Android 5.0--- is different from before

# Hardware Abstraction Layer

- Standard interface between software APIs and hardware drivers
- Two Components:
  - Module: HAL implementations
    - Shared library modules (.so) files
    - Dynamically linked
  - Device: abstracts the actual hardware




Implement functionality without modify higher levels.
The hardware abstraction layer (HAL) defines a standard interface for hardware vendors to implement and allows Android to be agnostic about lower-level driver implementations.

Storage devices are not managed by HAL HOTPLUG

https://source.android.com/devices/ png: Hardware abstraction layer (HAL) components

(.so file). It contains metadata such as the version, name, and author of the module, which helps Android find and load it correctly.

An audio module can contain a primary audio device, a USB audio device, or a Bluetooth A2DP audio device. A device is represented by the hw_device_t struct. Like a module, each type of device defines a more-detailed version of the generic hw_device_t that contains function pointers for specific features of the hardware.

The hardware abstraction layer (HAL) defines a standard interface for hardware vendors to implement and allows Android to be agnostic about lower-level driver implementations.

# Libraries and Android Runtime

- Libraries contains frameworks for web browser (webkit), database (SQLite), multimedia, libc...
- Android Runtime environment
  - Core libraries
  - Android Runtime(predecessor Dalvik): the managed runtime used by applications and some system services on Android.
    - Execute the Dalvik Executable format and Dex bytecode specification.
    - Ahead-of-time compiled runtime
    - Modern garbage collection
    - Development and debugging improvements (Support for sampling profiler; more debugging features;  improved diagnostic detail in exceptions and crash reports)

Core libraries: Dalvik VM Specific Libraries/Java Interoperability Libraries/Android Libraries

# Android Framework

- A set of services that collectively form the environment in which Android applications run and are managed; toolkit for applications

- Includes the following key services:
  - **Activity Manager -** Manages application lifecycle and activity stack
  - **Content Providers -** Allows applications to publish and share data with other applications
  - **Resource Manager** - Provides access to non-code embedded resources such as strings, layout
  - **Notifications Manager** – Allows applications to display alerts and notifications to the user.
  - **View System** – An extensible set of views used to create application user interfaces.
  - **Package Manager -** Maintains information on the available applications on the device.
  - **Telephony Manager** – Provides information about the telephony services available
  - **Windows Manager -** Performs window management
  - **Location Manager** – Provides access to the location services; GPS, Android Network provider

## Android vs. Linux Introduction

- An effort initiated by Google to develop an OS for mobile devices
- It is based on Linux kernel
- Device drivers, network stack, power management and other core OS features are taken from Linux
- Libraries added to enable applications to control the hardware more efficiently
- It have a different Java Runtime Engine specifically designed for limited resource in mobile devices

Is Android a Linux distribution?
The short answer is no. Android is based on the Linux kernel, but is not actually purely a “Linux distribution”. A standard Linux distribution has a native windowing system, glibc and some standard utilities. It does not have a layer of abstraction between the user applications and the libraries. In general, it simply looks like this:

# Android vs. Linux Core Differences

- Target Architecture
  - Android is designed for mobile devices (ARM ISA)
  - Linux for general purpose processors (x86 ISA)
- Kernel Modifications
  - Android added alarm driver, power management wrapper, inter processor communication Interface, low memory killer, and kernel debugger and logger
- Standard C Library
  - Android implements a custom implementation of C library which is proper for low memory

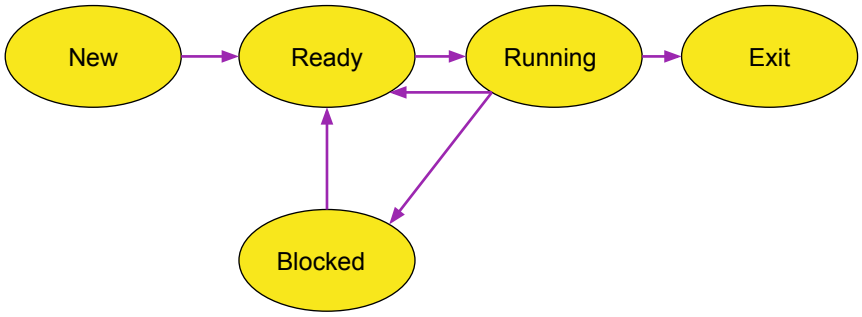
# Android vs. Linux Core Differences

- Dalvik Virtual Machine
  - A custom Java VM that simplifies the default Linux JVM
- File System
  - Android uses YAFFS (Yet Another Flash File System), Linux uses Ext, FAT, NTFS
- Power Management
  - Android is designed for saving battery in mobile devices. More about Android PM in the next “Power Management” Section

# Process Management *Overview*

Process Control Block(PCB) and Process Management

| Process state |
| --- |
| Process number |
| Program counter |
| Registers |
| Memory limits |
| List of Open files |
| ... |

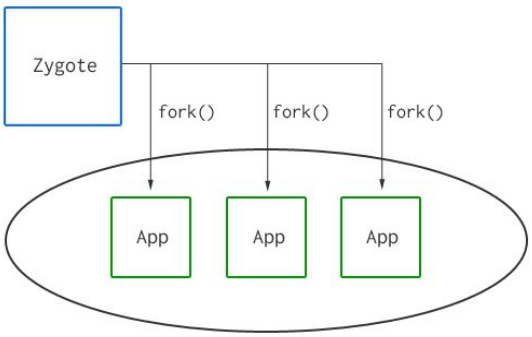



At base level android PCB is the same and PCB is managed by standard process management

# Process Management *Zygote*

**Zygote**: "It is the initial cell formed when a new organism is produced"

- Parent of all application processes
- Started by ***init*** when Android is started
- Have core libraries linked
- Improve launch time performance
  - Preload frequently used libraries




Zygote is a daemon service. All application processes are forked from zygote. Compare to creating VM separately for each app process, it improves the performance. All app processes can share VM memory and framework layer resources.

# Process Management Principles

Two basic principles for Process Management:

- Android OS tries to maintain an application process for as long as possible
- Only kills process when out of memory.
  - Remove process to reclaim memory for more important/new processes
  - Process to be killed is based on priority level. Processes are placed in “importance hierarchy”

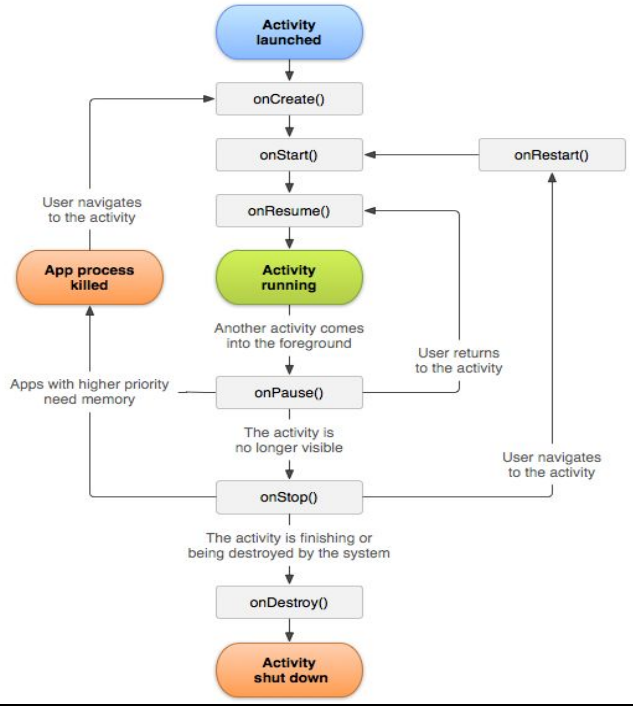
The two principles are reflected in the Process Life Cycle model

# Process Management | Application lifecycle 1

- Lifetime of an application process is not directly controlled by application, but rather by process management principles

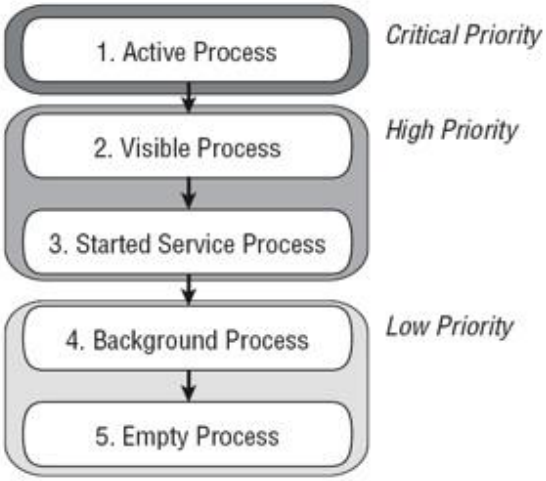
- Application lifecycle model gives user the multi-task experience
  - Easy to run multiple applications
  - Easy to switch between tasks

# Process Management
# Application lifecycle 2



# Process Management | Priority Hierarchy 1

- Linux reclaim memory by removing process with lowest priority
- There are five levels in priority hierarchy
- Android ranks a process the highest level it can



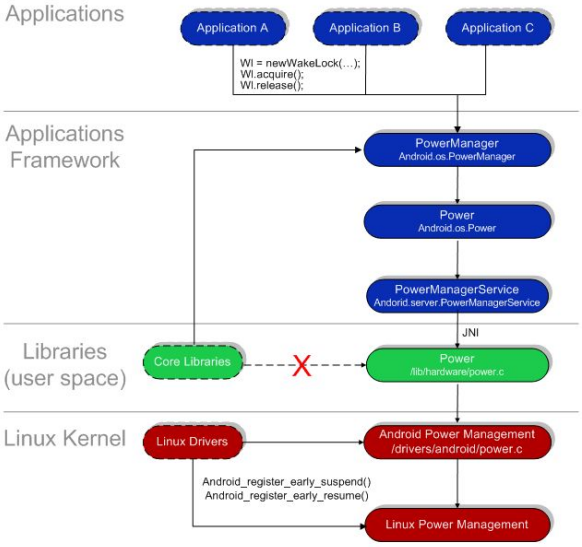

Android ranks a process at the highest level it can, based upon the importance of currently active components. For example, if a process hosts a service and a visible activity, the process is ranked as a visible process, not a service process.

# Process Management | Priority Hierarchy

- **Foreground process**: A process that is required for what the user is currently doing
- **Visible process:** A process that doesn't have any foreground components, but still can affect what the user sees on screen
- **Service process:** A process that is running a service. Started with startService() method
- **Background process:** A process holding an activity that's not currently visible to the user (the activity's onStop() method has been called)
- **Empty process:** A process that doesn't hold any active application components. Only alive for caching purposes

# Power Management Overview

- Android power management extends linux PM
- More aggressive power management policy than Linux PM
- Android Power Management Application Framework
  - A java application on top of kernel
  - Act like a driver that application can do power management using that




Android Power Management Application Framework is a java application that sits on top of Linux Kernel power management framework and act like a driver for power management. The JNI interface allows application to communicate with the framework

# Power Management | Kernel PM

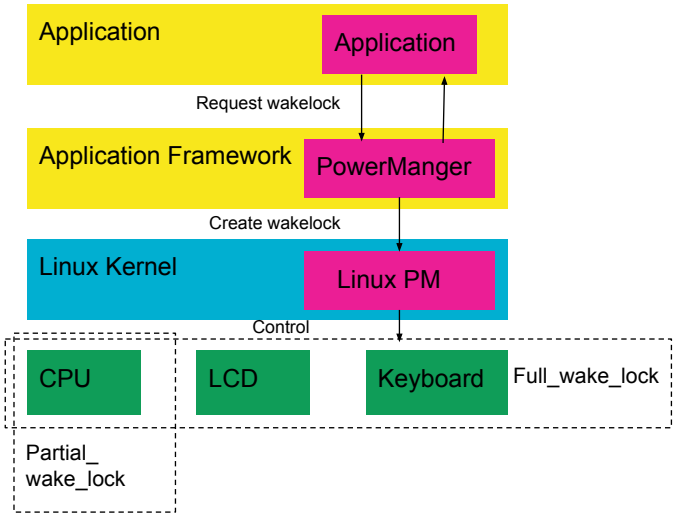
- ACPI: Specification for power management in OS
- Different power states
  - E.g. for CPU
    - P States: Active performance states
      - Higher P states means lower performance:
        - Po: highest frequency, highest power consumption
    - C States: Idle states
      - Co: idle
      - C1: halt clock gated state
      - C3: sleep
      - C6: off
  - In Android it is critical to have power states for devices
    - screen , keyboard, wifi, etc

The deeper the P state is, the lower the power consumption is at the expense of lower performance. For example, a core in P0 state operates at a V/F point that offers the maximum sustainable performance under a thermal design power (TDP) constraint. Besides, as a part of P states, the processors support turbo (or T) states where cores can operate at higher V/F points than P0 state as they operate below maximum power, current, and temperature specifications.

C0, C1, C3, and C6 states denote idle, halt, sleep,and off states. The deeper the C state is, the lower the power consumption is at the expense of higher performance penalty due to longer wake-up latency.

# Power Management

- Device goes to sleep when timeout
- Process must obtain wakelock to keep the device awake
- Different types of wakelocks
  - Partial_wake_lock
  - Screen_dim_wake_lock
  - Screen_bright_wake_lock
  - Full_wake_lock




The lightest wakelock should be obtained to maximize power saving

# References


1. Maker, Frank, and Y-H. Chan. "A survey on android vs. linux." *University of California* (2009): 1-10.
2. [Online]. Available: http://www.acpi.info/.
3. V. Pallipadi, L. Shaohua and B. Adam, "cpuidle: Donothing, efficiently," Proceedings of the Linux Symposium
4. Wikipedia, https://en.wikipedia.org/wiki/Android_%280perating_system%29
5. Android Open Source Project, https://source.android.com/index.html
6. CMU Distributed System Slides, http://www.andrew.cmu.edu/course/95-702/slides/Android.pdf
7. Tutorialpoints, http://www.tutorialspoint.com/android/android_architecture.htm
8. Android Studio Development Essentials: Android 6 Edition, Neil Smyth
9. Android App development, [online], http://rest-term.com/technote/index.php/Android
10. Android Kernel, [online] http://allenlsy.com/android-kernel-3/
11. Android developer guide, [online], http://developer.android.com/guide/components/processes-and-threads.html
12. Matt Hsu, Jim Huang,oxlab, "Power management from linux kernel to Android", 2010

# Questions?