Ziad Ibrahim Elbatal

2205075

G1

# **ANDROID OS**

**INTRODUCTION:**

An operating system mainly for mobile phones based on a modified version of Linux kernel and other open source , it was designed for touchscreen devices like mobile phones tablets and now is used for devices like TVs, smartwatches, etc. found and developed by GOOGLE ,it was revealed it in November 2007 and launched in September 2008.

it is free and open-source software. Its source code is Android Open Source Project (AOSP), primarily licensed ununderder the Apache License. Android was introduced with the Apache v2 open-source license,that allows many kinds of os to be developed for devices ,such as consoles, cameras, smart screens,etc.Most of the devices come preinstalled with a suite of proprietary software, such as Google Maps, YouTube, Google Chrome and Gmail.

**HISTORY:**

Android began as a startup company called Android.inc in 2003.the company was originally set out to develop an operating system for digital cameras but abandoned those efforts to focus on the broader market. Google then bought Android with its best employees in 2005 for 50 million dollars. Google marketed the early mobile platform to handset manufacturers and mobile carriers with its major benefits as flexibility and upgradability. Google was developing Android OS when Apple released the first iPhone ever in 2007. The prototypes of the Android phone was close to the Blackberry, not a touchscreen but with a physical keyboard. The launch of the iPhone changed the mobile computing market significantly and forced Android creators to support touchscreens more heavily. Until the HTC used ANDROID to be the first commercially available smartphone to run Android OS, featured a QWERTY keyboard and was met with some critical reception during its 2008 release. In the end of the 2007  the Open Handset Alliance  (OHA)

announced the devices that they are featuring ,including Google, Qualcomm, Broadcom, HTC, Intel, Samsung, Motorola . The alliance’s  goal was to contribute to the development of the first open source platform for mobile devices. Google then released the first public beta version Android 1.0 for developers at the same time of the alliance’s announcement in November 2007,then they released v1.5 in April 2009 introducing their dessert naming schema which was “Cupcake”. Around the time of the release of Android 4.4 KitKat, Google explained the story behind these names which was this statement “Since these devices make our lives so sweet, each Android version is named after a dessert."

**ANDROID Architecture:**

The Architecture divided into 6 layers: 1.The Linux Kerne: the lowest level of the Android architecture

2. Hardware Abstraction Layer (HAL): provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework.

3. Native C/C++ Libraries: Many core system components and services, are built from native code that require native libraries written in C and C++

4. Android Runtime (ART): Each app runs in its own process and with its own instance of the Android Runtime (ART)

5. Java API Framework: The entire features of the Android OS is available to you through the APIs written in Java

6. System Apps: the core apps with no special status among the user apps that comes with Android os like : email, SMS messaging, calendars, internet browsing, contacts, and more.

**1.ANDROID Process Management**:

Overview:

An operating system that's open-source, has a smart system for managing tasks. Its main aim is to make sure your device works well and responds quickly, regardless of what type of device you're using. Task management in Android involves important jobs like scheduling, assigning memory, and handling resources. Let's explore these key aspects further.

The Process Lifecycle Hierarchy:

Android has 5 different states at any given time ,from the most important process to the least important :

1. Foreground process
2. visible process
3. Service process
4. Background process
5. Empty process

Inter Process Communication (IPC):

Since Android is based on Linux, we can use Linux techniques like network sockets and shared ﬁles. Android system functionality for IPC is Intent, Binder or Messenger with a Service, and Broadcast Receiver. The Android IPC mechanisms allow us to verify the identity of the application connecting to your IPC and set security policy foreach IPC mechanism.

There are 2 ways for the process to communicate with one another:

1. Using Intents: An intent is to perform an action on the screen. It is mostly used to start an activity, send a broadcast receiver, start services and send messages between two activities.

Intent intent = new Intent(this, SomeActivity.class);intent.putExtra("key", "some serialisable data");SENDING MESSAGE

startActivity(intent);

1. Using Android Interface Deﬁnition Language (AIDL): It allows us to deﬁne the programming interface that both the client and service agree upon in order to communicate with each other using interprocess communication (IPC).

Comparing the android os with the IOS we will se that there is differences in the process management like:

1.android os is more flexible than the IOS

2.android uses a priority-based scheduler to manage tasks but IOS uses a deterministic scheduler

3.in android It employs a less restrictive approach to memory management, while in IOS Memory management is more stringent

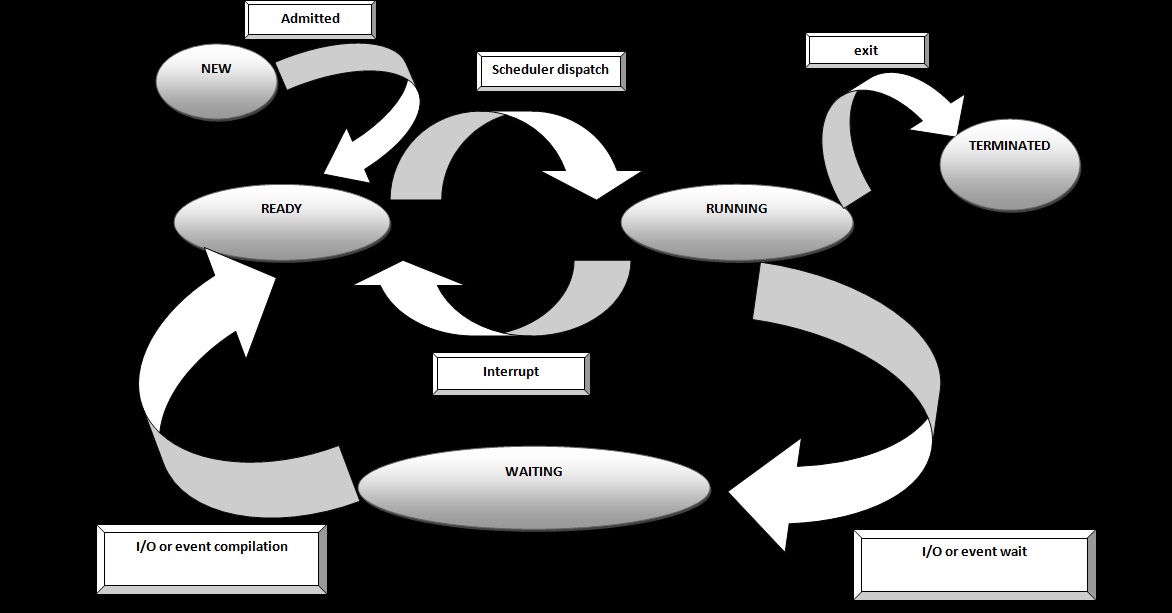
**2.Process States:**

**As mentioned earlier these are the process states :**

1. Foreground process: The app you’re using is considered the foreground process. Other processes can also be considered foreground processes
2. visible process: A visible process isn’t in the foreground, but is still affecting what you see on your screen, like when you open an app on your phone. Let's say you tap on the icon for a game. The process of launching that game involves several visible steps.
3. Service process: A service process isn’t tied to any app that’s visible on your screen. However, it’s doing something in the background.
4. Background process: Background processes are not currently visible to the user. They have no impact on the experience of using the phone. At any given time, many background processes are currently running. They’re kept in memory so you can quickly resume using them when you go back to them, but they aren’t using valuable CPU time or other non-memory resources.
5. Empty process: An empty process doesn’t contain any app data anymore. It may be kept around for caching purposes to speed up app launches later, or the system may kill it as necessary.

A thread has the following states:

* Blocked: The thread is blocked and waiting for a lock
* New: The thread has been created, but has never been started. A thread is started by calling its start () method.
* Runnable: The thread may be run.
* Terminated: The thread has been terminated.
* Timed waiting: The thread is waiting for a specified amount of time.
* Waiting: The thread is waiting.



**3.Cpu Scheduling**

Android uses o(1) scheduling algorithm as it is based on

Linux Kernel 2.6 Therefore it is named as Completely Fair Scheduler(CFS) as the processes can schedule within a constant amount of time, regardless of how many processes are running on the operating system.

Android uses two different scheduling classes

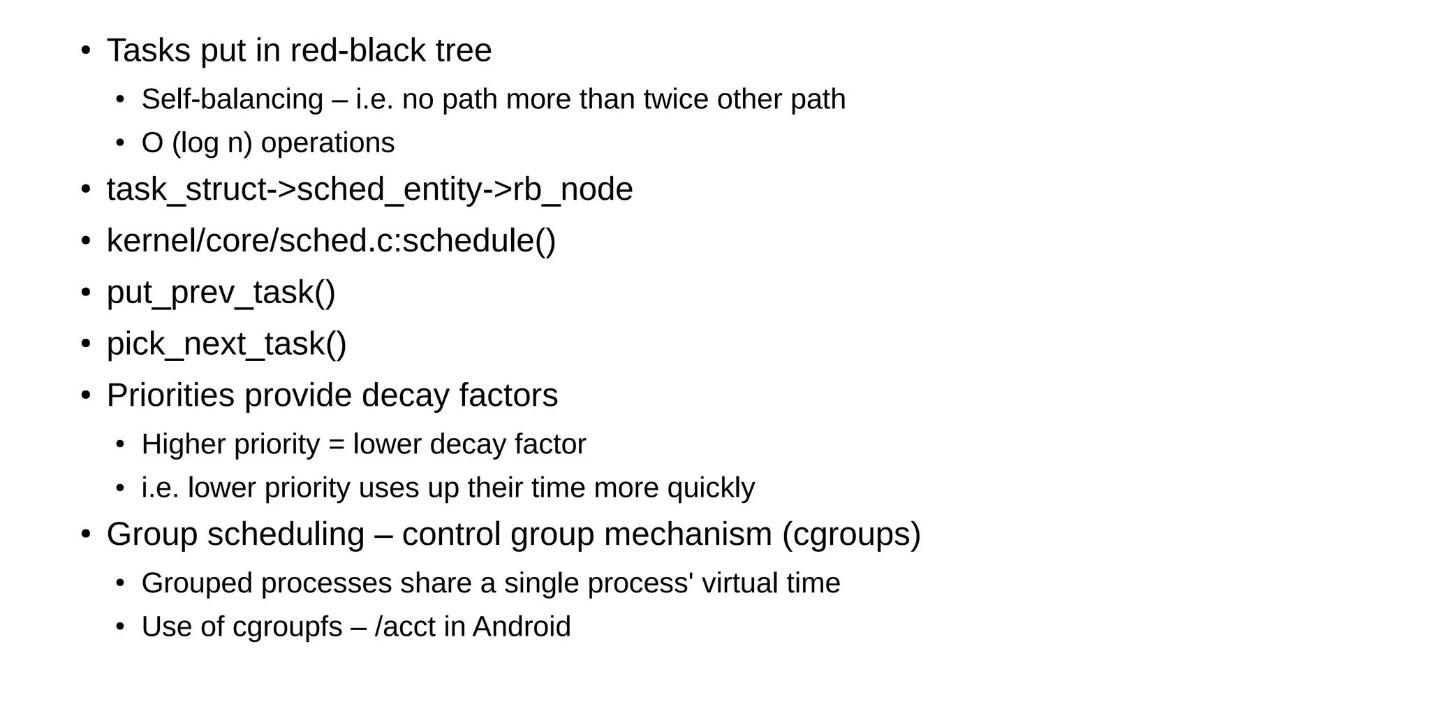
1.Background

2.Foreground

• Background class is low priority and can maximum utilize -5% of the CPU

• Foreground task has better priority and can utilize -95% of the CPU.

• We can change the priority by calling Thread.setPriority that is part of the standard Java API and contains a value from MIN\_PRIORITY(1) to MAX\_PRIORITY(10).

CFS:

Analysis of scheduling criteria:

In the Android operating system, efficient scheduling is critical to ensuring a responsive user experience, optimal resource usage, and effective power management. Here’s an in-depth analysis of the key scheduling criteria: throughput, turnaround time, waiting time, and response time.

**Throughput**: refers to the number of processes that complete their execution in a given time frame. High throughput implies that the system can handle more tasks efficiently.

**Turnaround** time: is the total time taken from the submission of a process to its completion. This includes all phases such as waiting, execution, and I/O operations.

**Waiting Time**: s the amount of time a process spends in the ready queue before getting CPU time.

**Response time**: is the time taken from when a request is submitted until the first response is produced. It is critical for interactive applications.

Comparing with ios:

both Android and iOS employ robust scheduling mechanisms to ensure high throughput, minimal turnaround time, reduced waiting time, and quick response time. However, they differ in their approaches and APIs:

**Android**: Provides explicit APIs like JobScheduler and WorkManager for managing background tasks and uses thread pools for concurrency. Developers need to manage concurrency and task scheduling explicitly.

**iOS**: Uses Grand Central Dispatch (GCD) and OperationQueue for efficient concurrency management, providing a more integrated and easier-to-use model for handling tasks and prioritizing execution.

Both platforms aim to optimize performance and user experience, but the tools and methods available to developers differ, reflecting the underlying design philosophies of Android and iOS.

**4.Deadlock Handling:**

Deadlock is a common platform in multiprocessing systems.

Deadlock reasons:

A deadlock occurs when a thread enters a waiting state because a required resource is held by another thread, which is also waiting for a resource held by the first thread. If the app's main thread is in this situation, ANRs are likely to happen.

Deadlock preventation:

There are majorly 4 conditions which if get satisfied then

we can prevent deadlock

Conditions are

1) Mutual Exclusion

2) Hold And Wait

3) No Preemption

4) Circular Wait

Deadlock avoidance can be done with Banker’s Algorithm

1. Banker’s Algorithm

It is a resource allocation and deadlock avoidance algorithm which tests all the

requests made by processes for resources , it checks for the safe state , if after

granting request system remains in the safe state it allows the request and if there is

no safe state it does not allow the request made by the process.

2. Preventing recursive locks

References:

<https://prezi.com/ebrz6qfjnjua/linux-android-scheduling/>

<https://arxiv.org/pdf/2104.09487>

<https://www.techtarget.com/searchmobilecomputing/definition/Android-OS>

<https://www.researchgate.net/publication/350992546_Android_OS_CASE_STUDY>

<https://developer.android.com/topic/performance/vitals/anr#:~:text=A%20deadlock%20occurs%20when%20a,ANRs%20are%20likely%20to%20happen>.