Android OS - Processes Management

Overview

– **Android process management** is similar to that of **Linux** at a low level,

– **But** the **Android Runtime** provides a layer of abstraction to help keep often used processes in memory as long as it can.

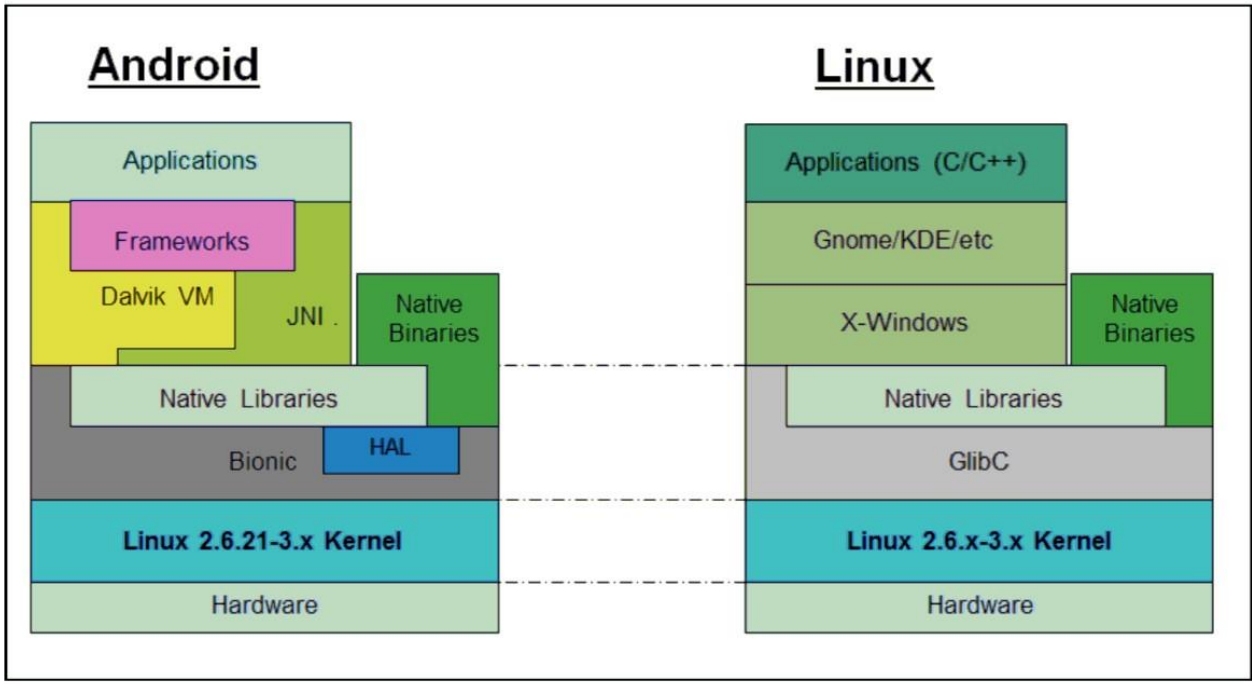
This is done using some **memory management techniques** that are not common.

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**.

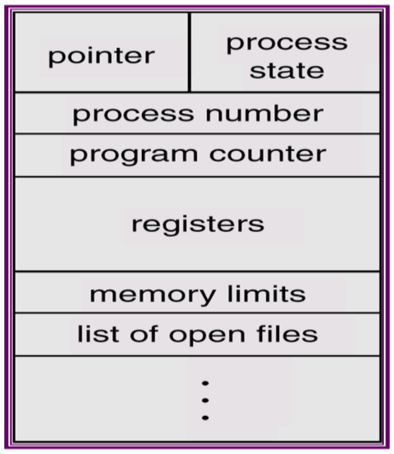


Android Processes

- A **process** is an **instance** of an **application** that **is currently running**.
- An **application** can have one or more **processes** associated with it.
- Android uses a **process-based approach** to manage applications.

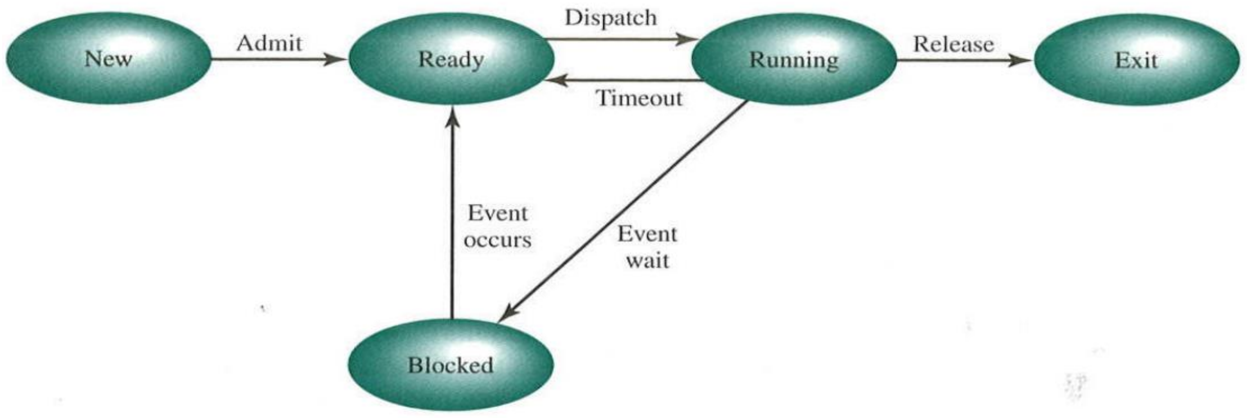
Process Management Overview

- Process management in a typical operating system involves many complex **data structures** and **algorithms**,
- **Android** is **similar** in that at the base level the **control structures** look the same.



Process Control Block (BCP)

This data structure is managed by a standard process management, which is something like this:



Android Applications

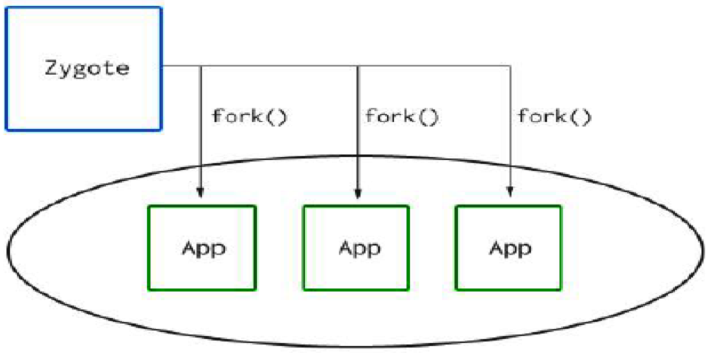
Android applications differ from standard applications in a couple very significant ways.

- Every android application:
  - Runs in a separate process,
  - Has its own Dalvik VM
  - The designers assign each application a unique UID at install time. This means the underlying Linux kernel can protect each applications files and memory without additional effort.
- There is no single entry point for android applications.
  - An **application** is a collection of components that can be used in other applications if desired.

Concept	Android Applications	Standard Applications
Underlying System	Dalvik Virtual Machine (DVM) or Android Runtime (ART)	Operating System (Windows, macOS)
Process Management	Process-based (multiple processes per app possible)	Single process per application
Security	Sandboxed with restricted access	More unrestricted access to system resources
Permissions	Require explicit user permissions for features	May not require granular permission control
Hardware Integration	Optimized for mobile features (touchscreen, GPS, accelerometer)	Not optimized for mobile features
Distribution	Primarily Google Play Store (with some sideloading)	Downloaded files or software provider website

Zygote

- Android at its **core** has a **process** they call the "**Zygote**", which starts up at **init**.
- It gets its name from dictionary definition: "It is the initial cell formed when a new organism is produced".
- This process is a "**Warmed-up**" process, which means it's a process that's been initialized and **has all the core libraries linked in**.
- When you **start an application**, the **Zygote** is **forked**, so now there are **2 VMs**.
- The **real speedup** is achieved by **NOT copying the shared libraries**.
- This memory will only be copied if the new process tries to modify it. This means that **all of the core libraries can exist in a single place** because they are read only.



Process Priority

Process priority can be set via the **Process.setThreadPriority**, At the base level, it uses the same process nice levels as Linux.

1. **Foreground process**: A process that is required for what the user is currently doing

    - It is **running** an Activity at the **top of the screen** that the user is interacting with (its onResume() method has been **called**).

    - It has a BroadcastReceiver that is **currently running** (its BroadcastReceiver.onReceive() method is executing).

- It has a Service that is **currently executing code** in one of its callbacks (Service.onCreate(), Service.onStart(), or Service.onDestroy()).

2. **Visible process**: A process that **doesn't have any foreground components**, but still can **affect** what the user sees on screen

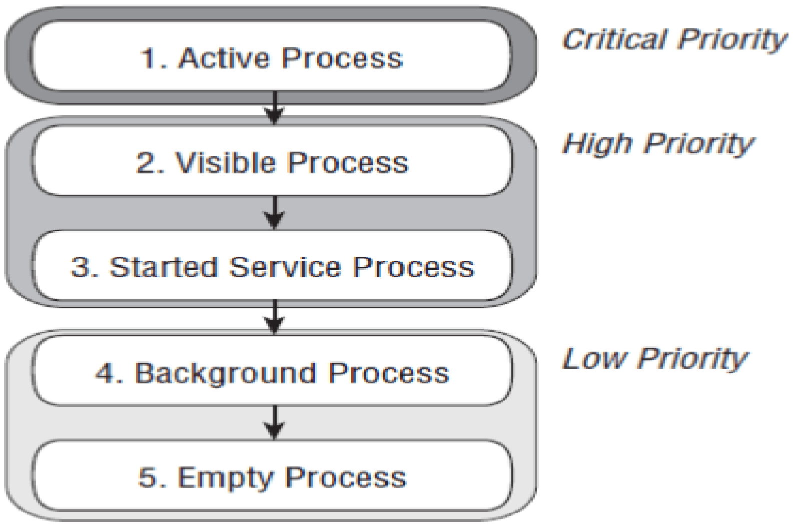
    - It is running an Activity that is **visible to the user on-screen** but **not** in the **foreground** (its onPause() method has been called). This may occur, **for example**, if the foreground Activity is displayed as a dialog that allows the previous Activity to be seen behind it.
    - It has a Service that is **running as a foreground** service, through Service.startForeground(), such as playing music, navigation, or a file download.
    - It is hosting a Service that the system is using for a **particular feature** that the user is aware, such as a **live wallpaper**, **input method service**, etc.

3. **Service process**: A process that is **running a service**. Started with startService() method. Such as

    - Background Data Synchronization: **EX**- sync application data with a serve
    - Playing music
    - Uploading/downloading data
    - Performing long-running operations

4. **Background process**: A process holding an Activity that's not currently visible to the user (the activity's onStop() method has been called)

5. **Cached process**: A process that **doesn't hold any active application components**. Only alive for caching purposes.



Process Priority

Process Termination

When does a process die?

Processes can be **killed** in a couple discrete ways.

1. An application can call a **method** to **kill** processes it has permission to **kill**.
   - If the process **isn't part of the same application**, it **can't kill** other processes.
   - On **install** you can actually **grant** an application **permission** to kill other applications.
2. The Android OS has **a** least recently used queue that keeps track of which applications **haven't been used**.
   - If the OS starts to run **out of memory**, it will **kill** the least recently used application.
   - There is also **priority** given to **applications** that a user is interacting with, or **background services** the user is interacting with.

Reasons for Termination:

- **Normal Completion**
- **Low Memory:** The Android system it might terminate background processes with lower priority.
- **User Action:** The user can explicitly terminate an application by swiping it away from the recent apps list.
- **Resource Abuse:** If an application consumes excessive resources (CPU, memory, battery) for an extended period, the system might terminate it to protect overall system performance.
- **ANR (Application Not Responding):** If an application becomes unresponsive for a prolonged duration, the system might terminate it to prevent a frozen user experience.