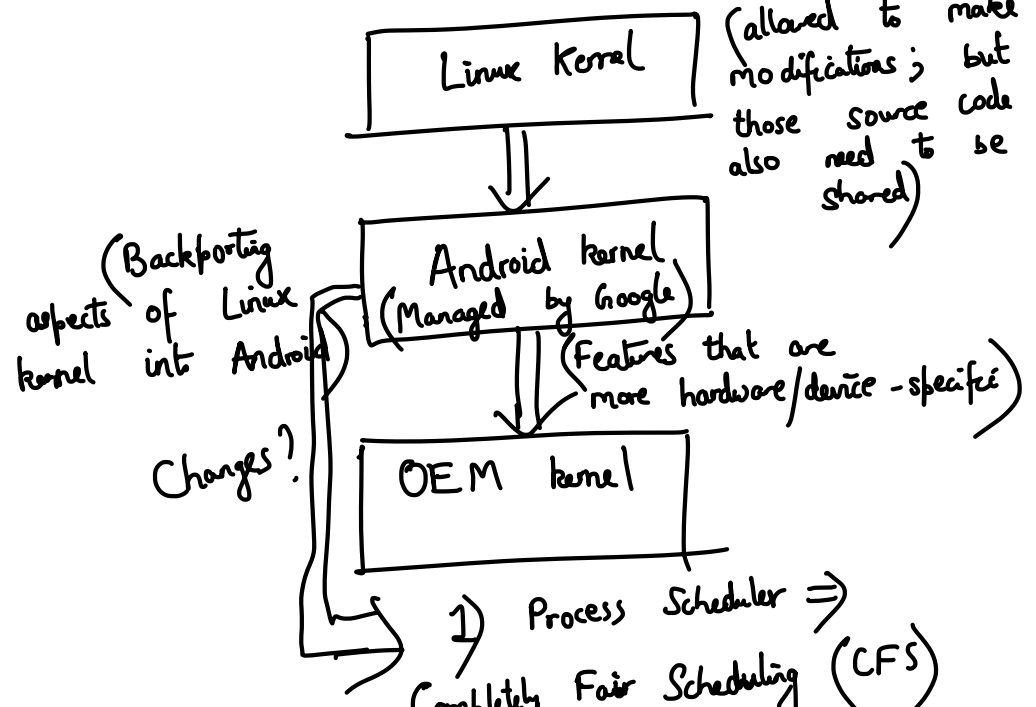
⇒ Requirements of desktop systems are often different from smartphones / mobile systems

— Requirement of fast response
— Hardware is different; more heterogeneity
⇒ Last day's picture
— Power constraints (Smartphone must adjust the functioning according to availability of battery)
— Security ⇒ Later

How does the Android kernel handle this?



1) Process Scheduler ⇒
Completely Fair Scheduling (CFS)
⇓
Not at all suited for mobile computing (Why)?
⇒

1) Asymmetric cores ⇒ [Big-Little cores]
[Not aware of power consumption]
Power is conserved more by executing on the big cores.

2) By default, unless a process specifies, equal time was allocated. So responsiveness was not given priority.
⇒ Foreground apps given higher priority.

New scheduling policy called Heterogeneous

Multi Processing.

⇒ Apps in the foreground
⇒ assigned to the bigger core.

Apps in the background
⇒ assigned to the little core

An important app can actually be run in the bigger core
⇒ By specifying CPU affinity.

Any difficulty of using this scheme?
⇒ Resource Environment created (Android 13)
⇒ which assigns Tokens to apps when a phone's battery is charged.
⇒ as an app keeps making various requests, its tokens are depleted and its requests get less priority.

Processor Requests

Is there any power management task being done on other hardware?

⇒ Once a request is sent by an app for some HW service, the HW tends to again sleep.

⇒ There is a specific type of lock called wakelock, where it is possible to keep the HW on. Again, can be a source of power depletion.

⇒ Apps that have unnecessary wakelocks and do not release the wakelocks on time can consume power.

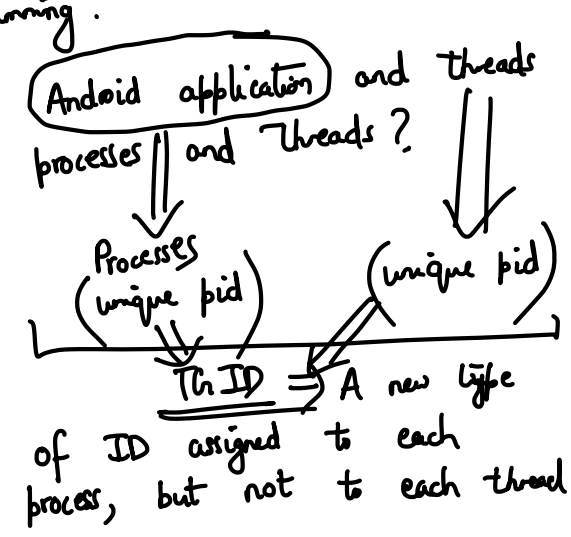
Memory Management

⇒ Smaller memory
⇒ Desktop: swap memory ✗ Mobile system
Smartphones: external storage = SSD
↓
limited no of writes allowed + a single change of bit requires changing the entire cell.

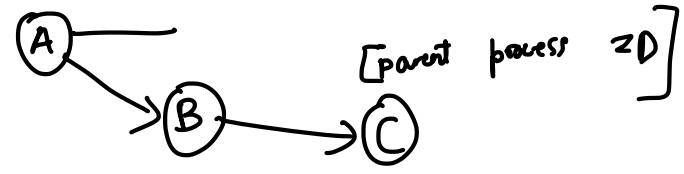
⇒ Android has an extra service that checks the amount of RAM available. If sufficient RAM is not available, then this service goes and kills the last used background process.
(OOM Killer)

Processes & Threads : ⇒ Every Android app uses threads. Because threads were an integral part of Java programming.

How does an (Android application) and threads map to Linux processes and threads?



⇒ A new type of ID assigned to each process, but not to each thread.



Asynchronous Programming

⇒ Different from the traditional programming

⇒ "Seamless" experience

⇒ Input is accepted whenever the user is ready to provide.

⇒ Processing of input starts as soon as resources are available, but need not end before the next input.

⇒ Works using signals [i.e. when an input arrives, a signal is sent to the process. The process execution jumps to a specific function.]

⇒ Javascript / Kotlin ⇒ more supportive of such [asynchronous programming].

⇓

Android has moved towards over the last 4-5 years.

Drawback ⇒ 1) Developers may find it harder to visualize

2) Debugging is more challenging.

Android ⇒ tries to mitigate the problem by using different ways of emulating sensor input.