MOBILE COMPUTING

Hardware Abstraction layer: An interface that connects hardware to RS

Ex: NFC

Daemons: Daemon is a computer program that runs a background process, rather than being under the direct control of an interactive user.
control internet

If there are 2 Network cards, 4 IP's can be assigned. Native daemons helps in connecting to IP address

Wi-fi BGN ⇒ physical layer

Volume Daemon : Disk control

Power Daemon

Init: Parent process of everything except what's running in Java/ Python kotlyn

Android Runtime:

Dalvik Virtual Machine: android Virtual machine optimised for mobile devices.
JVM drains out the battery if run on android

Zygote: responsible for warming up the system's cache and starting the System Server
Special daemon whose job is to launch apps.

Android isolation ??? depends on runtime

Android keeps a copy of jvm bytecode
Zygote provides a virtual machine at runtime

monolith vs microkernels

Apache Harmony: (not in use anymore)
(↳ implements java APIs / Ex: System.out processes

System service

surface flinger → updates VI

Activity Manager → manages activity life cycle

File system in userspace:

Binder: Component Object Model. like a HashMap → Transferring using AIDL
Android Interface Definition Language.

JNI - Java Native Interface

An object appears from one process to another across VM boundaries

Zygote vs init

java c++

Keychain process: OS stores keys in keystore that contains different private, public keys.

key encapsulation: only a specific UID can retrieve the private key of that process

Isolation is found at the process level:

A program running in VM is also inside a sandbox. whats happening in the network basically outside is blocked. ∴ To communicate we require IPC, APIs

In android binder does that job
↳ not linux

binder is the brain of android
↳ does all input management

Input manager communicate with android apps through binder

Binders take care when there are crashes, sends signals.

Binder does memory mapping

FP stdout 1 stdin 0 — mmap: brings in

Binders can transfer files easily. Just that file.

Binder notes down owner UID, caller UID, grants permission on just the object being called.

Binder space gives only 1MB space to each process

Binder keeps tracks of all file sharing (bookkeeping)

WindowsManager in system server manages the windows we see which actually in turn the binder manages.

A parcel is a virtual file and handle the file to a process by parsing the FD.

mmap: map a region into a process without allocating an actual memory. we can mmap only a FILE.

Ashmem: Android shared memory

It can grow and shrink. It can pin and unpin memory regions.

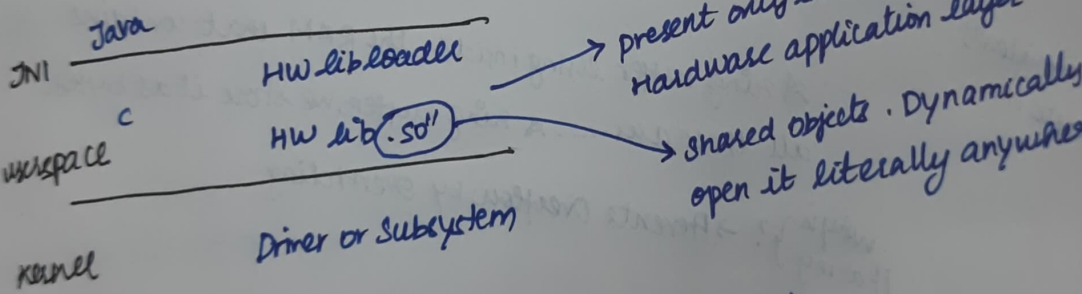
A virtual file to be parsed is created using ash memory.

Both mmaps and window manager creates shared memory. (Binder ain't here)

A FD is passed from process to process through binder. If a process dies, the file goes to binder.

pmem: Physical shared memory → not used by apps. used by underlying hardware services.

Apps
↓
Android
↕ Binder
System Server
+HW related service



If application wants to load an API, it dynamically loads.

SQLite queries dynamically load using shared objects

Hardware can implement APIs & put it in shared memory & give it to companies.

When a particular android is installed, it may not have certain functions: not supported problems occur upon android updates.

They disable certain functionalities

Ex: noise cancellation

Ex: Camera: while on a VC, you can't snap. All these are handled.

wakelogs

sleep, wake → unlock

↳ everytime you lock

↳ different in android than in unix

Ex: Screen does not sleep when on VC but other

islog
logbuff
logger → has main buffer, system, event, radio

↗ every application (main buffer)
↗ sys server initializes (system)
→ android creates and sends to the other OS to communicate with antenna (radio)

OS → android
OS → (2 runs at the same time)

Why is called a buffer?

read and write is fixed. That's why phones slow down

we cannot just insert the log files into filesystem, it slows down the phone.

storing strings are expensive because similarity we store logs as byte in buffer.

A ring buffer living inside the RAM that contains all these are.... A ring buffers we store it as bytes.

why is it a ring }? → Prevents overflow by overriding.

Apache license ← protection not from end users

ASC: You can't use google if you use android

Hardware Abstraction Layer:

In linux, we gotta manually add in codes. In android HAL is shared.

How is a view get passed?

AF_INET

DGRAM
STREAM
RAW

2 linux commands
→ ioctl
→ netlink

IOCTL: a raw interface. May work for one device driver and not work for another.
RAW socket. usually bytes, but also character
can implement MTCP, QUIC, SCTP using RAW

old version of firewall is done by netlink, now it is ioctl.

ioctl is the base for every system call.

Netlink is a unix domain socket without any files. It has protocols
- RTM (management of routing tables)
- Firewall
- NFLOM, etc

netlink knows how to send responses, to where using PID.

FHS (File Heirarchial standard)

EX: /root/user ; /etc, /bin Android does not used this

Android uses a different root systems

Android uses: /system → immutable while the OS is running. the mount point cannot be changed. Read only.
Remount

to reboot in android

ext4 file system

↓

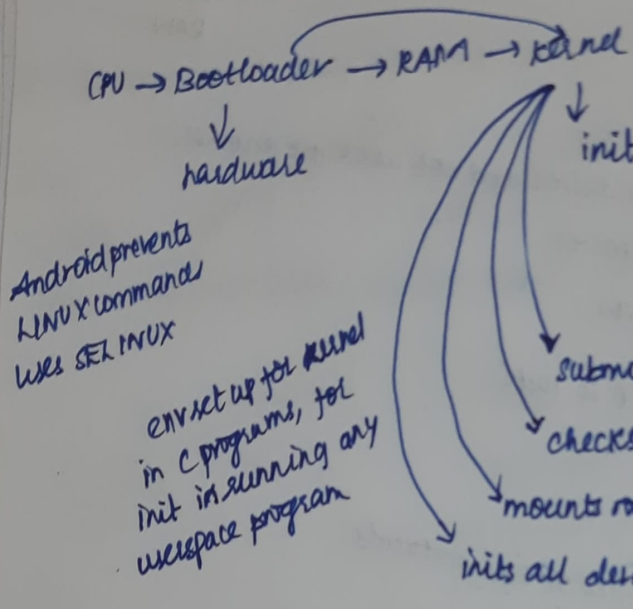
/data → mutable / read & write

/data/stardo → SD cards are in /data.

When the computer is switched on, bootloader is searched for. It initializes RAM
The bootloader expects certain hardwares to be present and it looks for them.

Ex: keyboards, c

Bootloader checks for hardware state



init.rc → a bunch of commands file to init to make it check for things required.

Init

→ loads native libraries
→ network files
→ env setup
→ starts daemons

→ Baseband processor communicates with devices.

looks into LD_LIBRARYPATH env variable → Microdaemons

If a daemon crashes, init is responsible to restart the daemon.

Zygote is the most important daemon for android. (process no 37)

↗
VANILLA JVM
- Loads all java classes
- Resources are loaded (only available in android)
  ↳ includes XML, HTML, basically the presentation layer

when a request to start a new app comes in, zygote forks itself.

Zygote has android's classes & resources that holds all the APIs needed. To reduce the cost of startup, loading all the classes, we use zygote (anything that is not javafile)

System Server:

Anything to be done in android should pass through system server.

→ inits all system services

→ Service Manager starts

↓

invokes binder

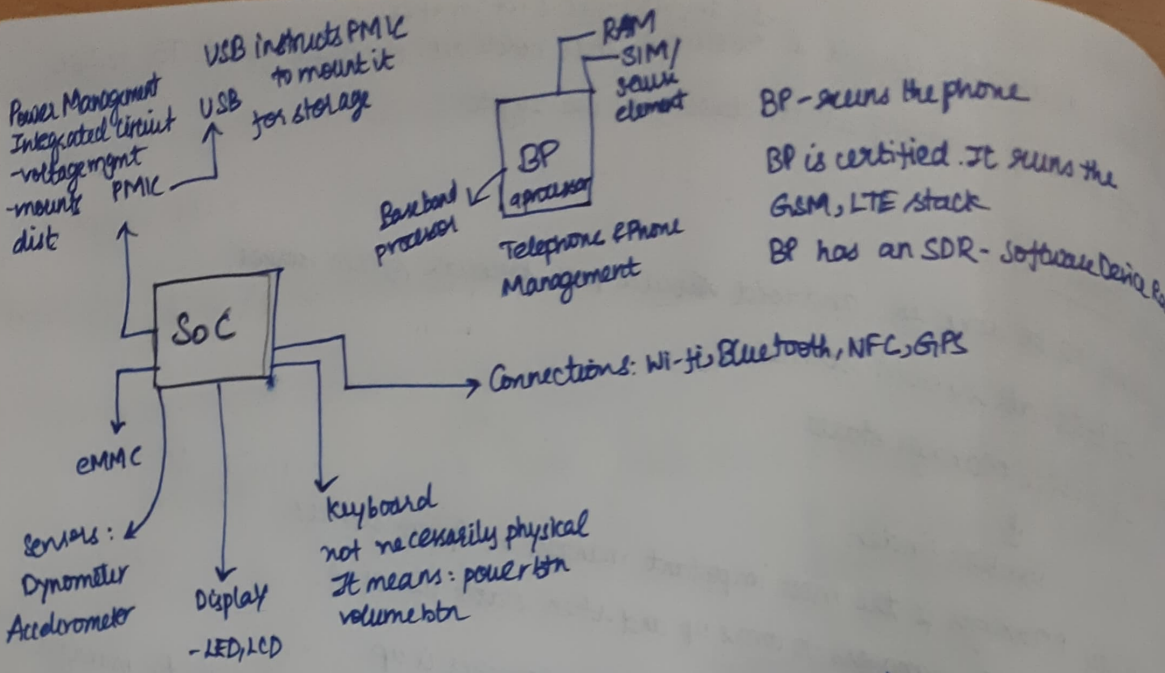
Activity Manager is the most important manager in System services.

→ waits for all managers to come up and then starts working.

Activity manager communicates with zygote and launcher is up.

Launcher tells activity manager to launch what app the user wants and activity manager intimates that to the zygote.

ueventD: /udev responsible to mount devices like SD card.



We don't use hard disk drives in mobiles as they have space constraint and moving parts

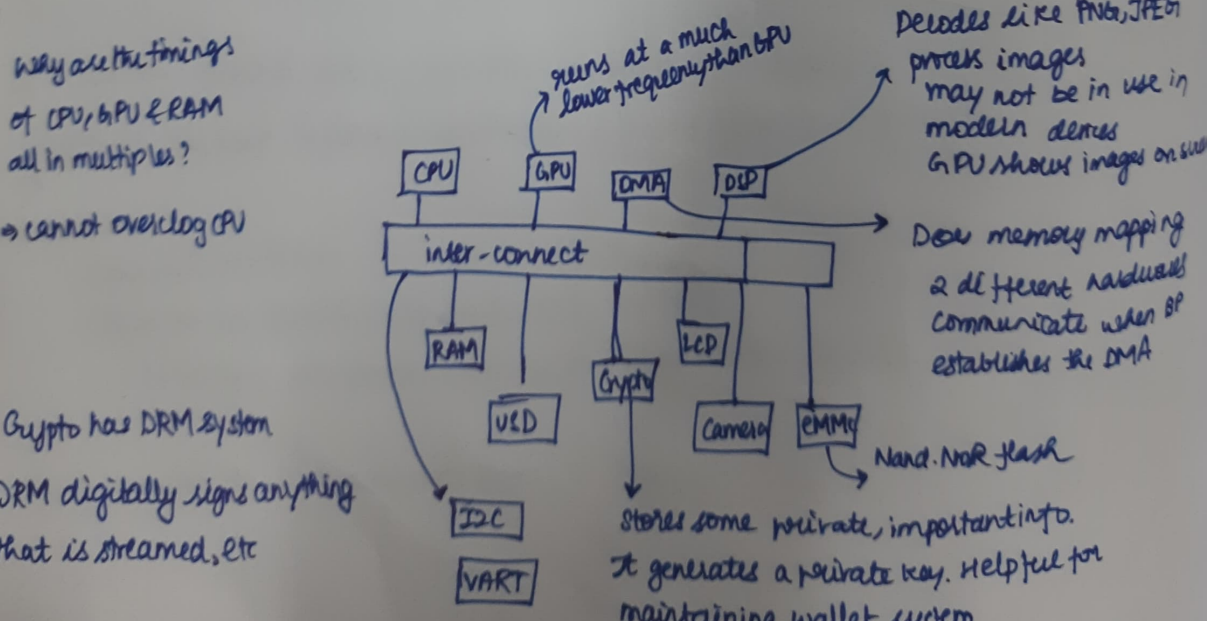
No SSDs in mobile as they are bigger in size

eMMC - Multi-media Card, a card that can be mounted to phones. The life of eMMC is too short. The flash drives were slow, so SD cards need to run apps again & again.

SIM fuses itself if anything tries to access the personal key.

Secure element should be secure because SIM card tracks some important data that identifies subscribies. It is isolated from SoC & BP. It is read only.

operations like PrK (personal key) & PIN only are possible.



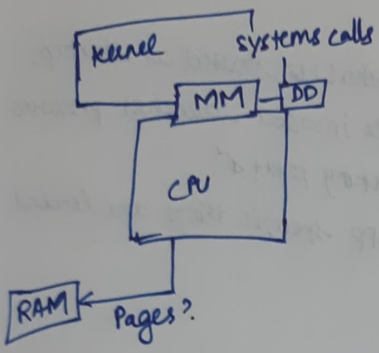
All passwords are stored in keystore. → crypto subsystem checks this?

I2C, UART, GPIO → Buses → widely used in android

↳ learn about I2C with one ex. look through docs

↳ communicates with power devices → similar to IOCTL. Through GPIO interface we can modify the spine.

similar to sockets



Bionic - android implementation of libc

libc.so is loaded into the system.

Sys call → libc.so load → kernel ↓ does operations ← open app?

CPU and GPU have clock ratio to improve frame rate, prevent resource starvation and prevent wearing of hardware devices.

SDR Software Designed Radio → if you want a device to be hackable.

→ open telephony a software for baseband restrictions.

BP runs realtime OS. Android runs realtime pipeline.

Monotonic time guarantees an increase time; you cannot decrease time

RTC - Real Time clock time can be changed; this is used in our mobiles which gets time reads from kernel → gets info from CPU

Monotonic time → needed in apps like encryption

RTD in BP is too fast because the kernel has multiple data structures in it.

↳ low latency audio

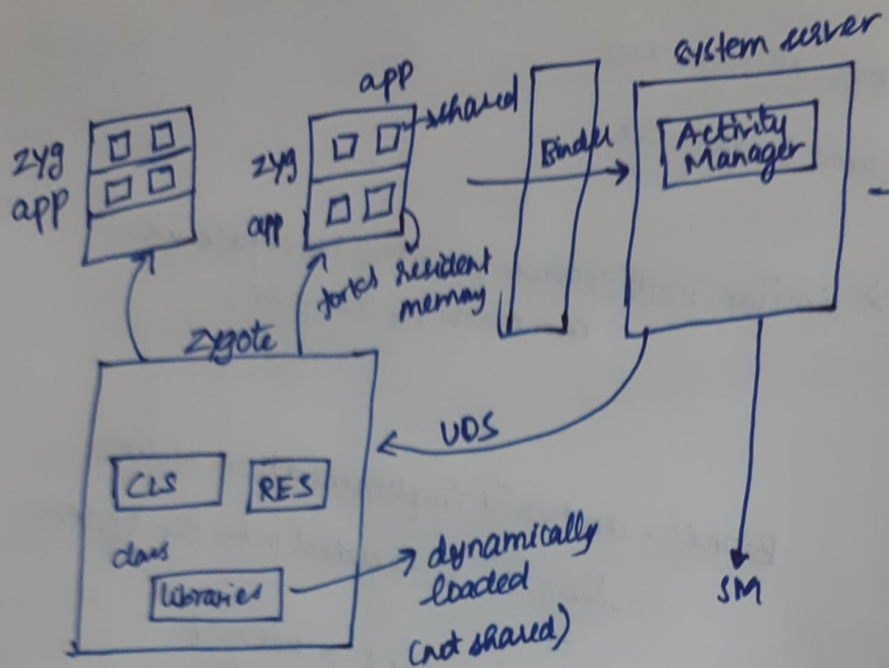
HAL - Audio uses RTD → This is how android guarantees RT processing.

LTE - sends voice packets out

BP is prone to hacking ∴ FCC didn't allow it to be along with SOC.

Graphic processing pipeline processes every 16 m/s. That's how it guarantees speed?

RTOS → only one main functionality



system up() { }
notifying AM is up. All the status of the children is broadcasted and checks which manager to wake up next.

launcher() is started as startup
zygote invokes launcher process
by forking itself &
then app specific things are loaded

Once the boot complete() is sent, apps can start themselves.

2 types of broadcasts

→ Onetime broadcast
→ Sticky broadcast

broadcasts to currently listening activities

EX: wifi - connections

→ once connected to network, broadcast occurs.