"Lecture 1"

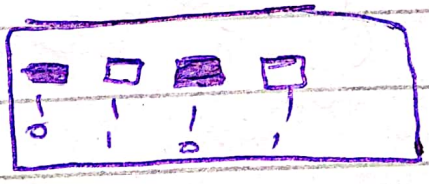
⇒ In early time, the computers were like a room size.

⇒ Toh us time programer apne customers se program leta tha, and usko convert karta tha "punch cards" mn and aur "punch cards" ko feed karta tha computer mn.

Punch cards ??

⇒ add al, 6 , iske equivalent ek card hota hai. ???

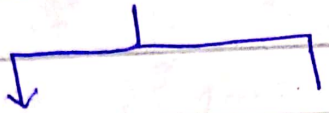
(i) Abb add al, 6 se jo binary banegi usko is card pe likhde



⇒ Abb humne kya kiya, agar humko 5 ka factorial calculate karna hai, toh hum uske liye, ek General code likhdenge,

⇒ And jab woh user se input mangega, toh us time, koi aur program, ye run kadega, its called reusability.

⇒ Job control.



// which will go first and second, that is its job ~~of the~~ control.

⇒ Multiprogramming

Keep several programs in memory at once and switch b/w them.

⇒ System Call

Hume agar koi cheez print karni hai toh, hum "printing" ka code nhi likhenge, hume jo cheez print karni hai sirf wohi, likhega, and

agar hum ye kaam khud karein toh its very difficult,

=> What we do, when we're printing we off-load it to somebody else, and this "somebody else" is operating system

=> System Call

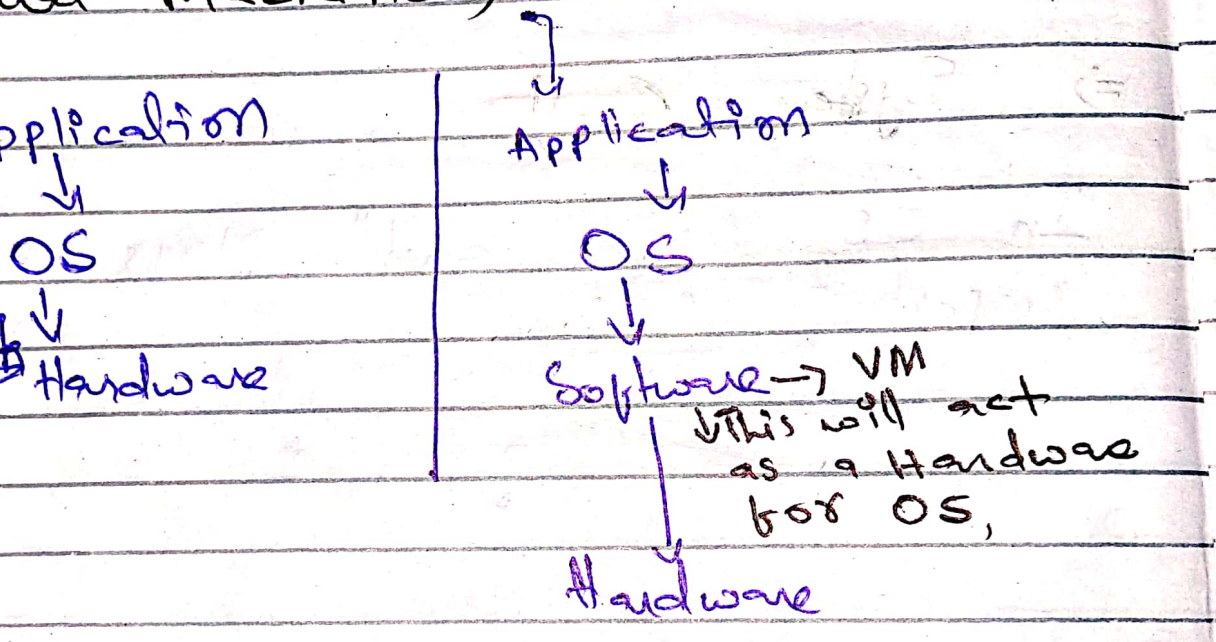
In OS "system call" plays an important role,

?

1970s: Unix

⇒ Portable operating system.
written in a high level language.

⇒ (Virtual Machines)



Application wants to talk with hardware. It directly don't, it first talk with OS, then the OS talk with hardware, and instead of that hardware we put the software that acts as a Hardware for OS and called Virtual Machine.

"Lecture 9"

Booting

- Boot loader

=> it starts the boot process.,

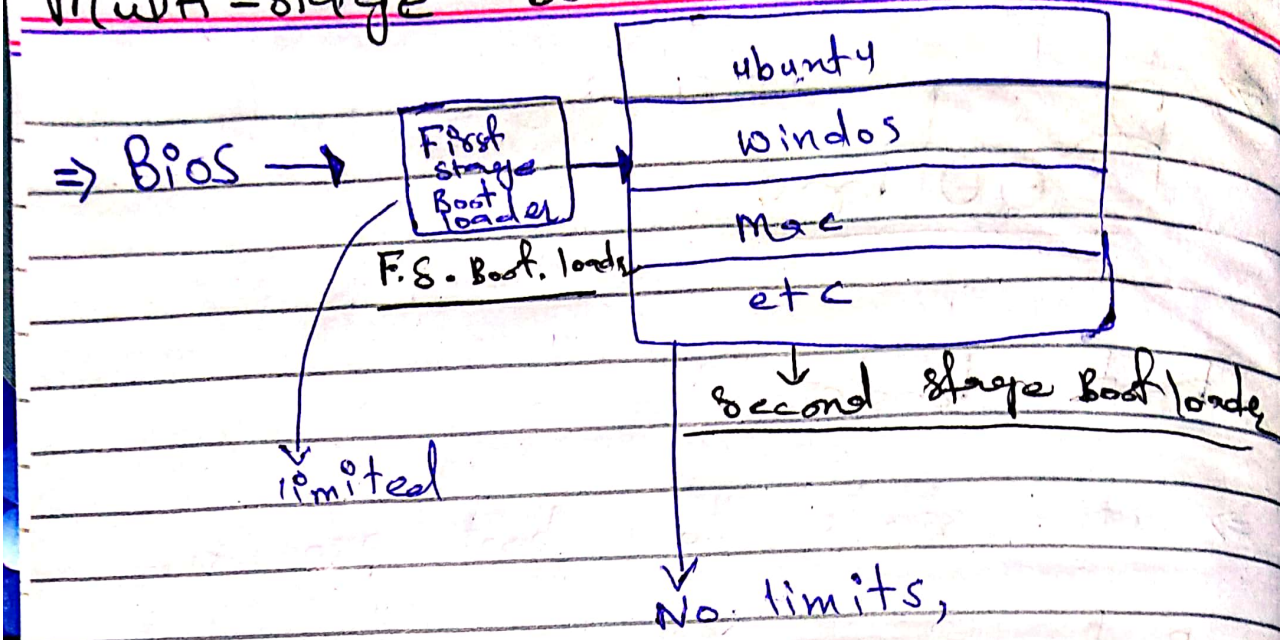
Boot process ?

=> operating system run hone se pehle jitni bhi cheezen aati hai woh boot process kehlati hain.

(i) Boot loader,

it will find that in which hard disk the OS is, and how many OS are in this hard disk that is the responsibility of boot loader.

Multi-stage boot loader



→ Booting Windows!!

Bios → OS → File system

~~⇒ File System dont know, how~~

⇒ BIOS don't know the FileSystem.

$ Hard disk

Everything, is in 0, 1, 0, 1 and we dont know which file is txt and which ~~is of another~~ py etc.

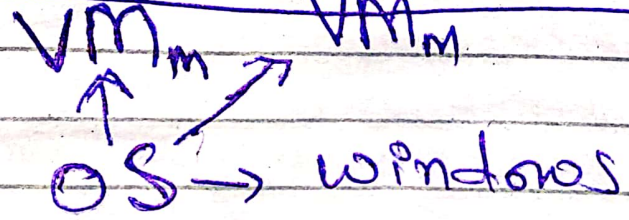
⇒ So "OS" handles this with the help of "File system"

MBR Master Boot Record

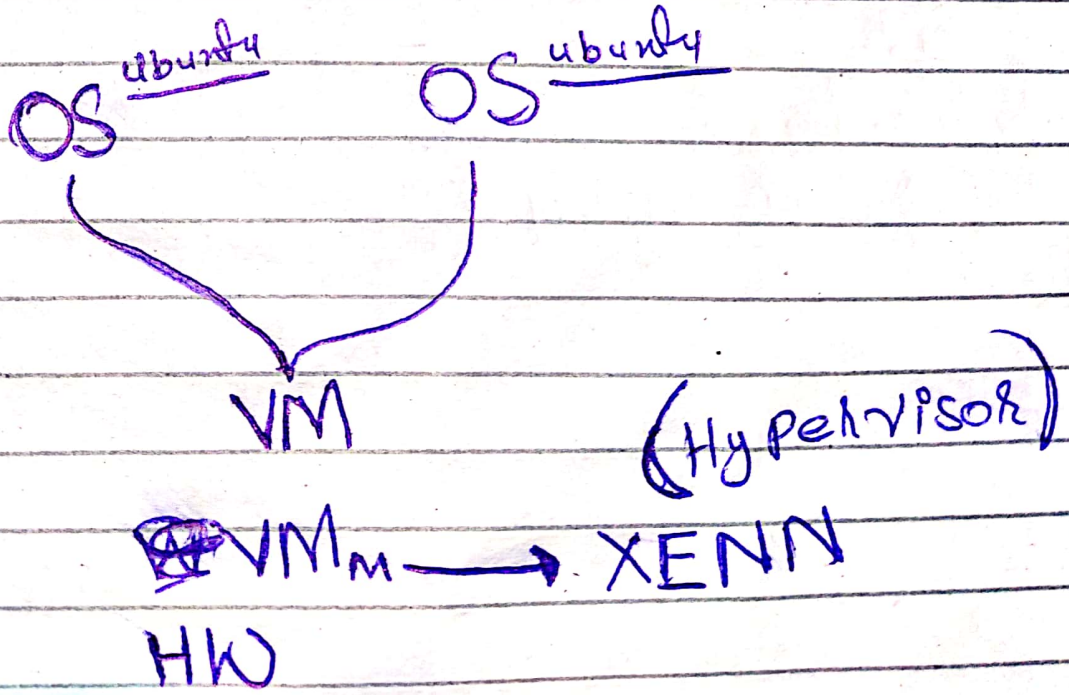
VBR

512 → How to boot ?

// Yahan pe humne beech mn windows daali hui hai, and uske baad 9 → ubuntus



⇒ High Performance



// And Now humne windows uda di hai and uski jaga ek hypervisor daala hai, ab kya hoga iski performance badh jaegi. :)

"Lecture 3"

⇒ Policies

What you will acheive is policy

⇒ Mechanism

How you will acheive ???

⇒ Processes

When we run the application is called process.

Definition of OS

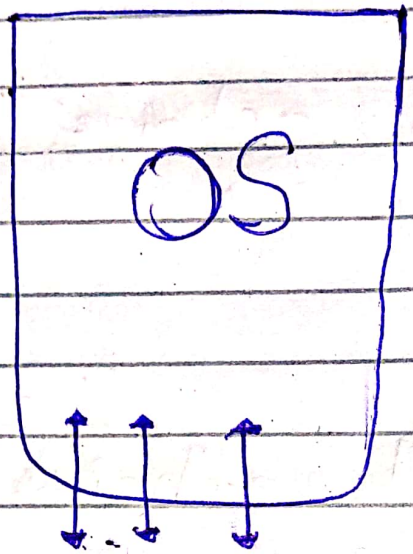
⇒ The first program
↓
, Not that good definition because first thing is boot loader,

⇒ A program that lets you run other programs.
↓
VM is also program, and also it doesn't fit that well.

"Architectures of OS"

~~Types of OS~~

(1) Monolithic



Monolithic

⇒ Everything is present in this one chunk Audio, video, in just one code ⇒ ek jaga pe koi "bug" ayega, toh saara code plagiam kharab hojaiga.

⇒ Everything compiled together.

⑤ Modular

⇒ We divide things into the modules

⇒ Har cheez ka apna module hogg

⇒ jis cheez ki zarooraat hogi sirf usa e load karengg, means dusri cheezon ko comment out kardengg.

chunk

Modular.

⇒ In monolithic and in Modular Everything runs in high previllage mode!.

(2) Micro kernel

⇒ High previllage

Do whatever you, want to do, nobody will ask anything.

⇒ Low previllage

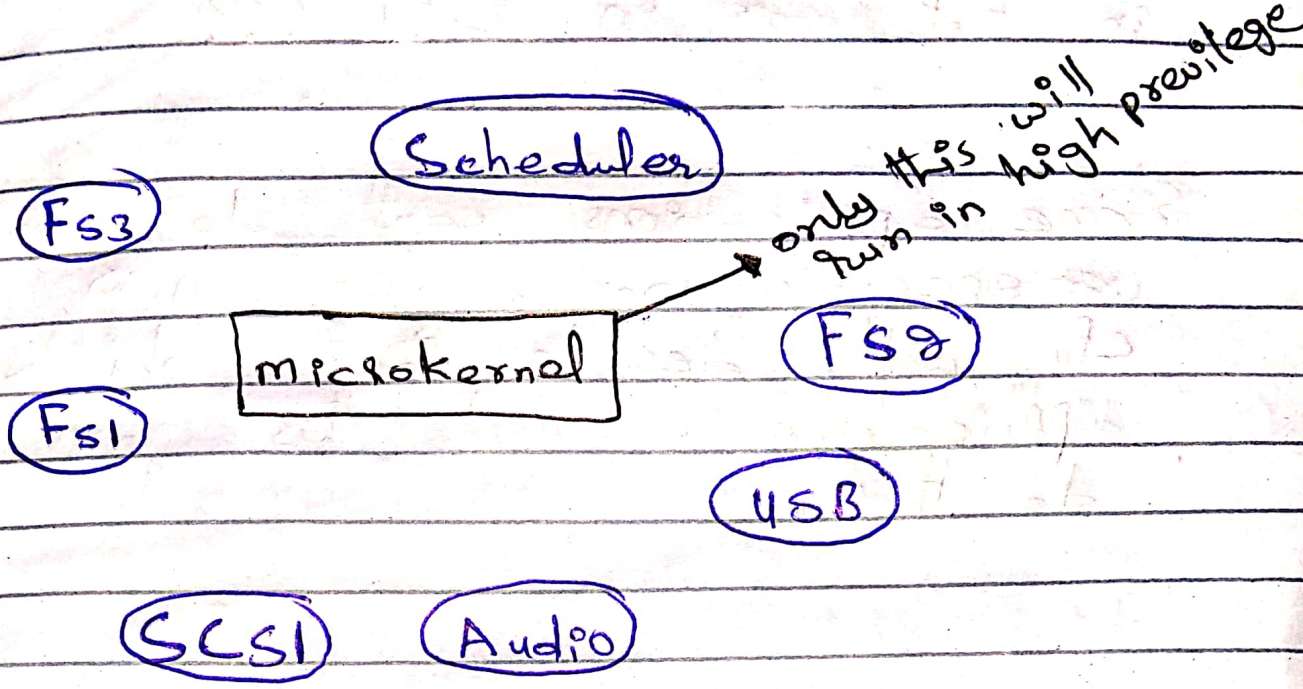
You can do everything, but there will be a check on you that, is this thing allowed to you or not.

(3) [illegible]

kernel

The part of OS which runs in ~~highest~~ high previlige is called kernel.

③ Microkernel



⇒ Sirf un cheezon ko high previlige mn rakhein, joh iske elawa kaam kar e nhi sakte.

⇒ And hum aun sari cheezon ko jama karte hain hain and name it "Microkernd".

⇒ Abb saari chezen high previlige mn run nhi hongi, sirf wohi hongi, jinke liye zaroori hai high previlige mn run hona.

Exp Unix kernel.

⇒ How you will know that its in H.P or in L.P ???

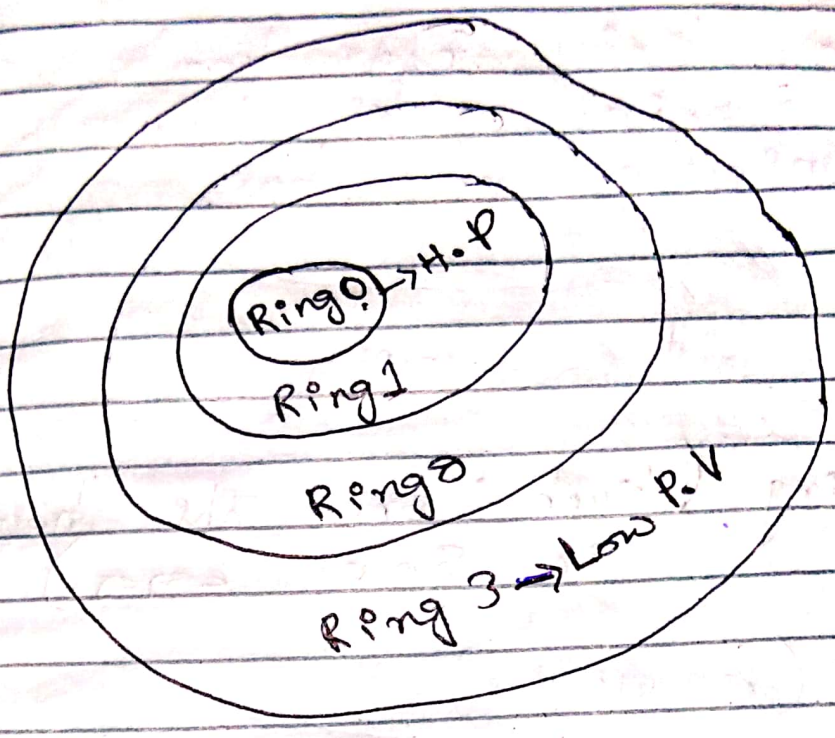
Some applications cant do some specific work. So how CPU will decide that which application will do this or do that.

⇒ Flag set karo agar flag "1" hai toh H.P wali saari instructions run hongi.

⇒ Agar flag "0" hai toh L.P wali saari run hongi.

⇒ Lekin agar flag "0" hai and H.P wali instructions run ho rahi hai toh usi tym CPU ruk jaega. :).

⇒ Iss flag ko change karne ki instruction H.P hai.



⇒ Whenever a L.P application try's to execute a H.P instruction. Hamara OS aura usko data hai, and hume control pass kardeta hai ke dekho ye kya kar raha hai.

⇒ Then we decide to give it a permission or not.

(Violations)

=> Hum koi aesa kaam kar rahe hein jiske hum q permission nhi hai,

Things will happen

(a) Nolting (kuch bhi nhi hoga in some cases.)

(b) Trap (exception)
=> hum ausa pakad lete hai

(i) Memory acess violation
(ii) Illegal instruction violation
(iii) Register access violation.

=> The OS processes to trap

=> Whenever you have a trap or exception, &

=> controll is passed to the OS.

=> Flag H.P kar deya jata hai. and controll OS ko pass kiya jata hai.

⇒ OS decide on course of action. → (Important).

⇒ Trap occurs

(i) Via software

(ii) Because of an access violation

(iii) Via a hardware interupt.

Interupt.

⇒ ek CPU mn sequence of instructions chal rahi hoi na pehle ye karo, ye karo, beech mn koi aesi cheez ajati hoi jiski wajah se ap iss sequence ko chod ke kahi aur jate ho (except jump) tha it is called interupt.

⇒ Aus tym CPU jo kaam kar raha hota hoi aur rokta hoi ~~and control OS ko transfer karta hoi~~ and ek trap issue karta hoi, and control OS ko deta hoi, and

OS ye decide karta hai
ki isko sath kya karna
and this is called (Interupt
handling). and uske baad
us application ko bolta hai
karo yeh karna hai.

Software Interupts (traps)

Packet → Network card ✗

⇒ We want to sent a packet
to the Network card.

⇒ We directly cant do that. we
need the help of OS

Paket ⟶ OS ⟶ Network card ✓

How OS will ~~sent~~ sent it ???

⇒ OS ne hume already batay
huwa hai if d want to
sent something to Network
card do this

```
mov eax, 10  and
int 80h
```

⇒ Abb jese e ye "int" dekhaga toh ek trap issue hojaigq. (Abb yeh chezen hongi after int)

(i) trap will be executed.

(ii) Previlige Flag "1" hojaigq.

(iii) controll OS ke pass chala jaegq. and this is called Mode Switch

⇒ Now After this OS will do this

(i) OS ne dekhq ean mn "IO" hai to it means network card mn packets sent karne hain

(ii) OS will sent the packets to the Network card.

(iii) And Again ye flag ko zero "0" pe set kardegq and controll application ko dedegq

⇒ When we go back from kernel mode to user mode ise called as Return from exception ⇒ Exam question

(System Call)

=> We use trap → int mechanism to switch to the kernel.

=> Pass a number that represent the OS service.

(i) syscall number : usually set in a register

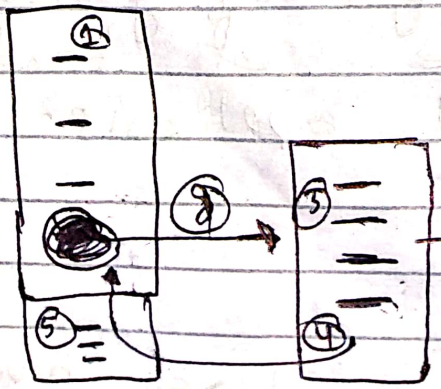
=> A system call involves.

(i) Set system call number

(ii) Save parameters.

(iii) issue the trap (into kernal mode)

- Os gets control
- Return from exception
  ↓
  Back to user mode
  ↳ Important.



System Calls

(i) Int → From Hardware.

(ii) ↓

- let suppose koi infinite loop chal raha hai and isi tarah koi application chal rahi hoti hai

- Application kisi ~~[illegible]~~ OS ko control nhi deti.

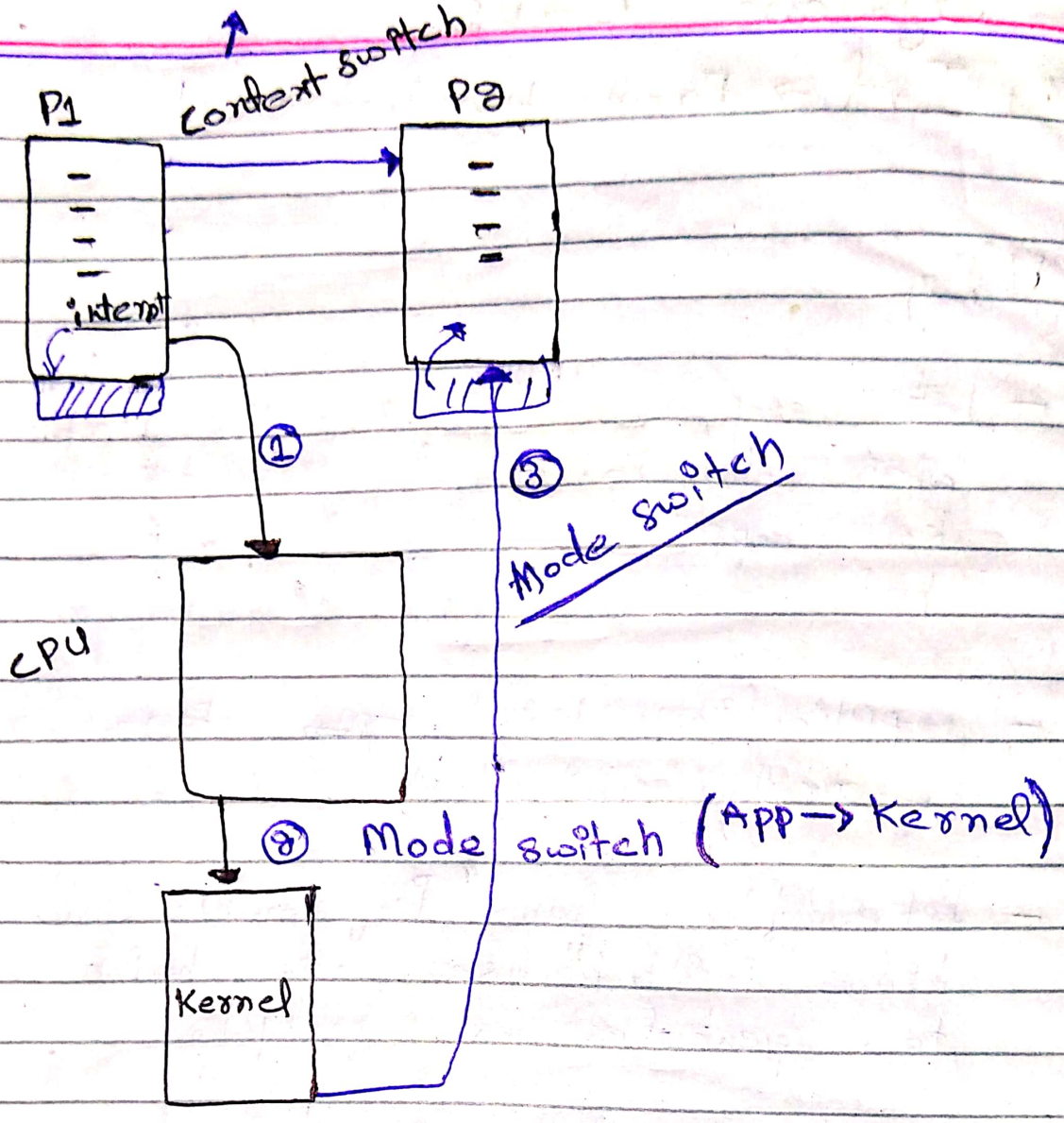
- Iske liye hum "Programable interval time 8254" use karte hain to generate an interupt.

- Ye kuch time ke baad automatically interupt issue kardeta hai.

- Since 2005, "High precision Event time" (HPET) replaces "8254 programable time"

(Yield) → Two cars on a road first → car will say you go first :)

2 mode switches in one context switch



1. P1 was running smoothly, then it got an interupt. and it dumped the registers of P1 in some where in the memory

2. And then it loads the registers of P2 from memory to the CPU and, done

switch

his work

(3) That is one mode switch

(4) Now it will again do the mode switch from P2 to kernel

(5) So Remember this from P1 to P2 (context switch) it includes two mode switches.

Main Points

1 context switch => 2 mode switches

Multiprograming

When More then one programs are running it is called as Multiprogramming. i).

Crural knowledge

(int) → very slow, used until 2005

(SYSENTER) → very Fast, 9x times faster than "int".

4 (Processes)

Downloaded Application ⇒ is it a application? or when it is running then its a application.?

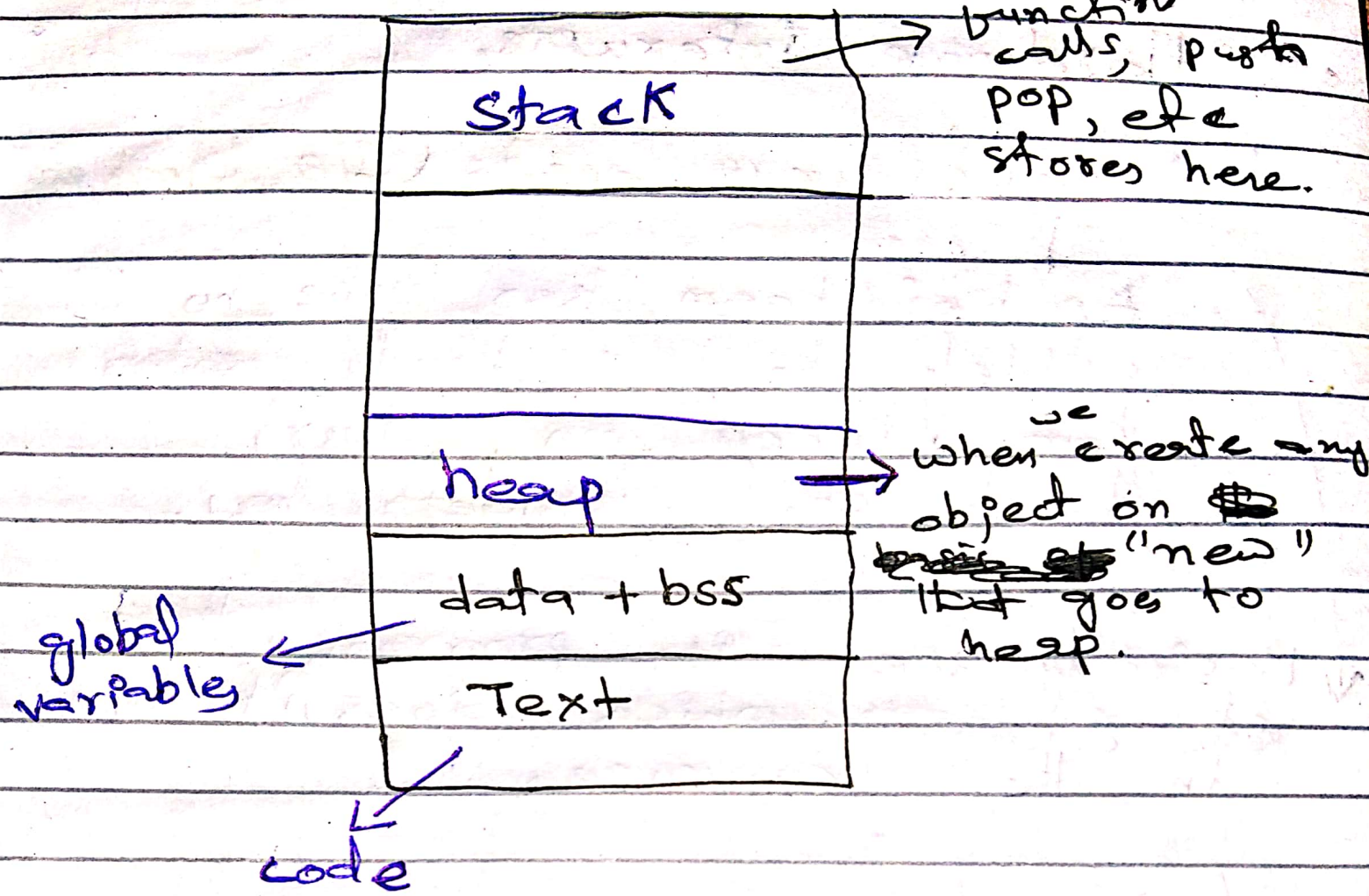
⇒ Potential to execute ⇒ Program

⇒ We it runs, or execution starts its called Process

⇒ When we Run a program the first thing OS does is that it gives the chunk of RAM to this Program.

⇒ Then OS loads the

Memory Mapping :- OS loads the executable file and puts it in the memory.



≠ Contexts

- Hardware interupts.

Asynchronus events (I/O, clock, etc

↓

Ap koi kaam kar rahe ho suddenly koi aur banda beech mn dusra kaam start kardeta hai.

↓ Basically we write program which doesn't run in the sequence.

- Software traps.

Process in a Multitasking Environ

- Multiple concurrent processes.
  Each has a unique identifier.
  Process ID (PID)

Exp MS word → PID = 8079

2nd tab
MS word → PID = 9019

etc ....

Multitasking.

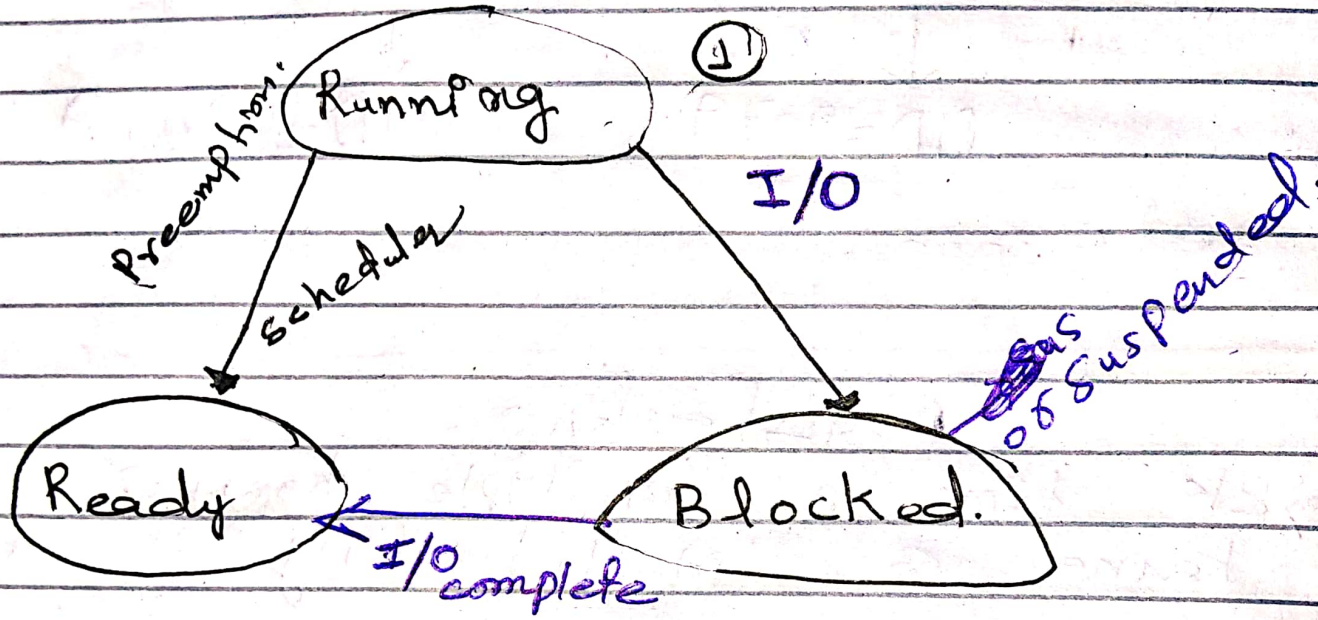
⇒ Ek time pe multiple kaam karna is Multitasking

⇒ Multiple kaam karna means that kabhi ye kabhi woh not all the work at the same time.

① OS loads the executable file and puts it in the memory.

② Now it is ready to run. But it will wait for its turn.

③ Process states



① Process is running. It issued some (I/O).

② Now this process will go to the "Blocked state"

③ When the process will done with the I/O. Then it will go to the ready state.

④ And it will wait until its getting the turn from OS

⑤ And when this (I/O) process is in blocked/Ready state at that time other process are running.

Scheduler:- Kis time pe konse process ko Ready se Running state mn lana hai ye scheduler ka kaam hota hai.

CPU-bound process → infinite loop → No I/O

Preemption:- OS ne apki execution stop kardi

⇒ let suppose infinite loop laga hua hai. on that time OS will switch your process to "Ready state".

⇒ Remain ye process block state mn nhi jaegi.

Process Control Block (PCB)

⇒ Har cheez ki information PCB mn store hoti hai it's like a struct in C

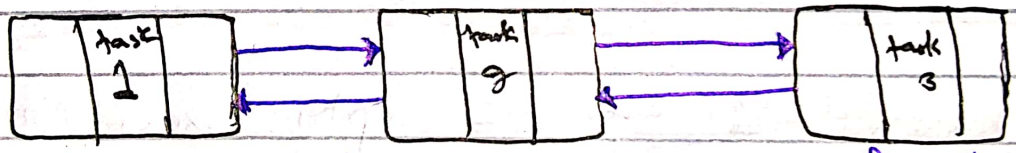
For exp

⇒ PID name

⇒ Process kaban se start huwa

⇒ Ye Process kitne time se execute horaha hai

Processes in linux.



⇒ Har process ka ek PCB banta hai.

⇒ Ye is mode ke andar PCB hoogi.

⇒ init → first process in linux

⇒ current → returns the address of the current running process.