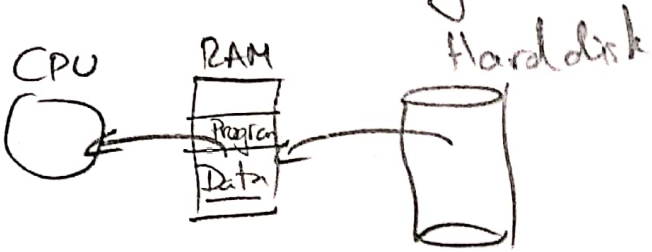
Operating Systems:

CPU → chip

→ A line in a program

Execuling of instruction in CPU

1) Memory → RAM
2) CPU → Hard-disk, Cache, Registers } Diff. memory registers.



HW + SW + Data

RAM → Program + Data

Why diff types of Memory?

Hard disk → Permanant
RAM → Non-Permanant } When switching off comp.

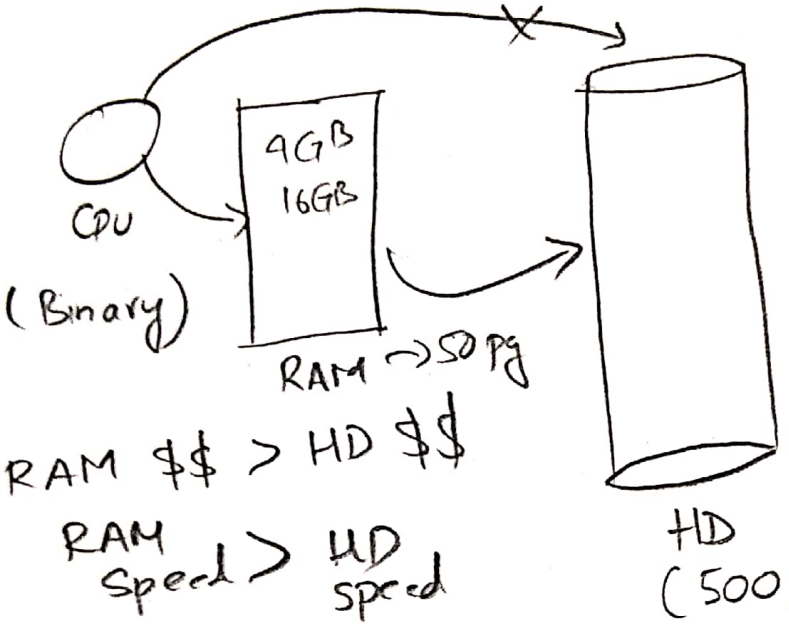
Fast
Slow

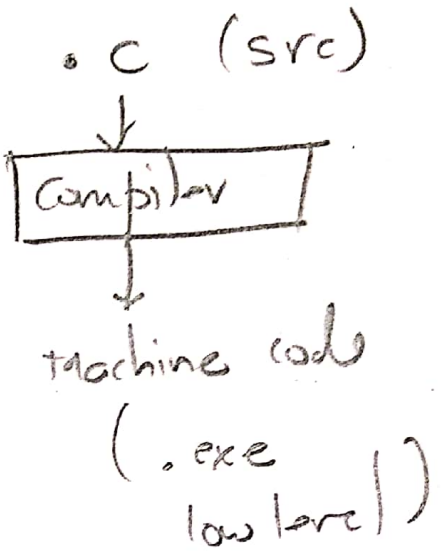
X HM

Ram size ∝ Speed

HD → Ram
Copy Paste
Data

} Speed + Size



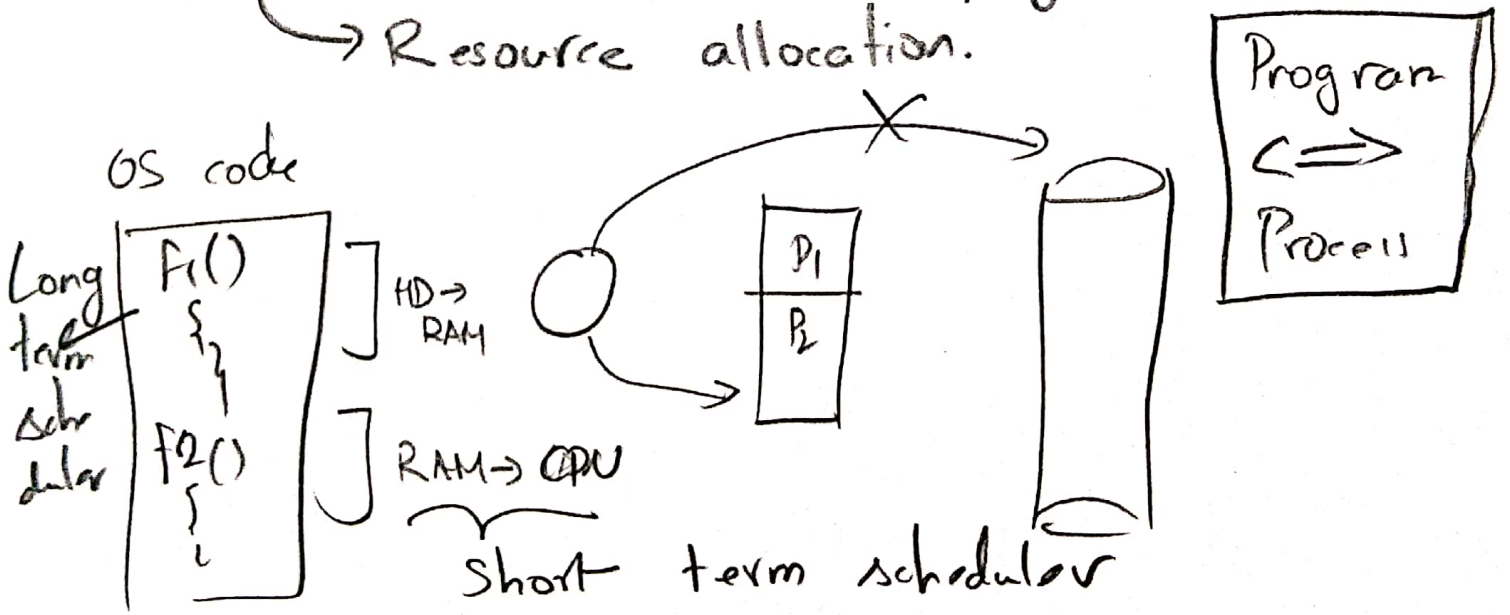


→ Compiler → also a software

Op system → Resource manager

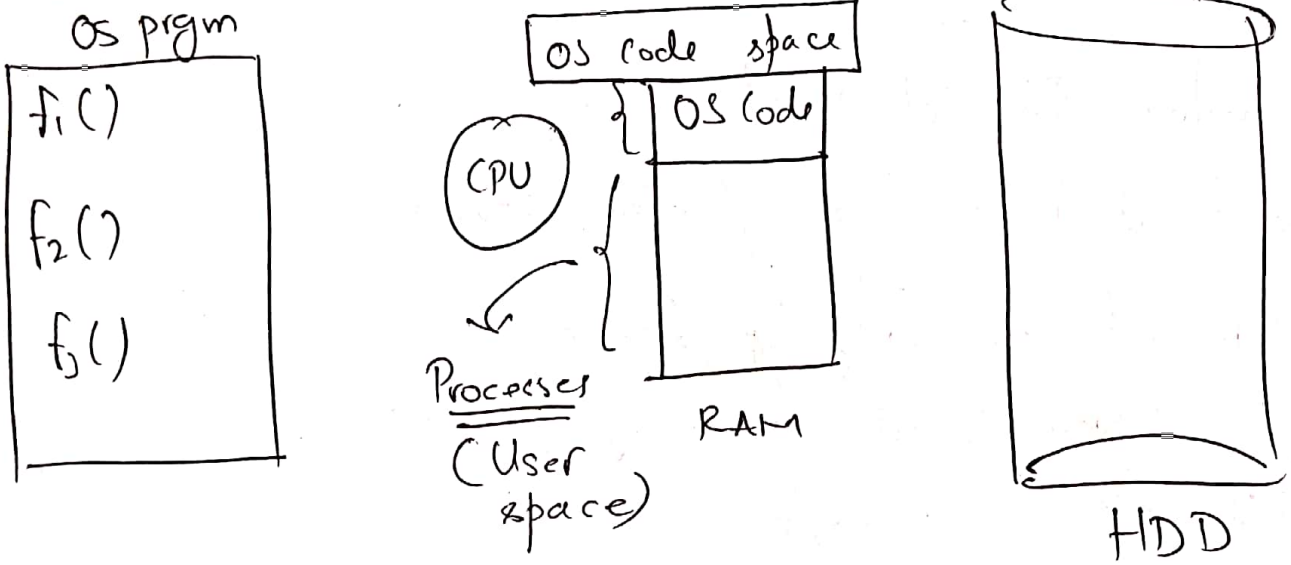
→ Predicting which programs are needed by CPU (done by OS)

OS → Long term scheduler (part of os)
selection of programs.
↳ Resource allocation.



⊗ Which processess should get it first?

⊗ Main fn of OS → Resource management
⊗ Long Term / short term scheduler.



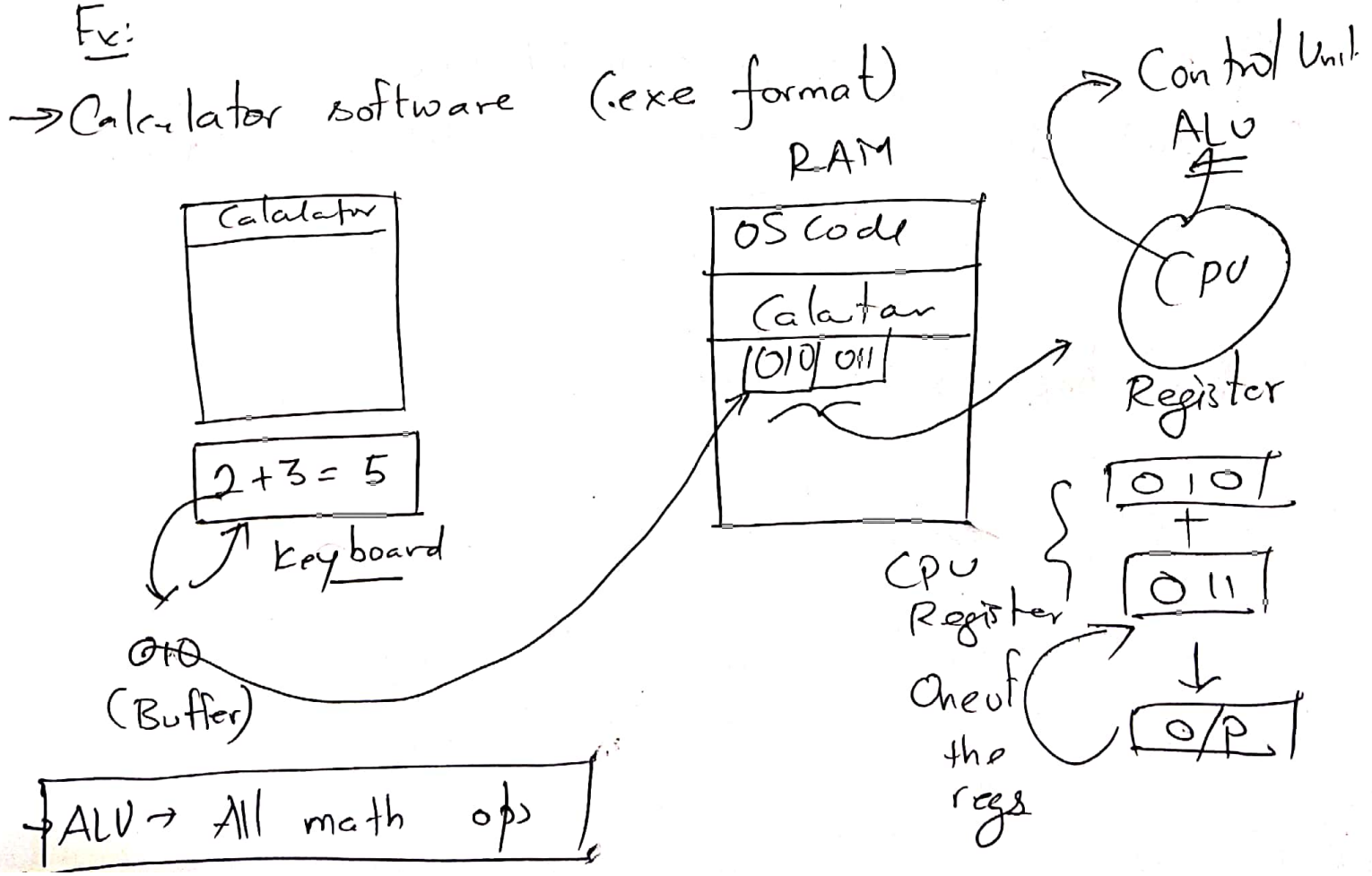
OS → always in RAM

I/O devices:

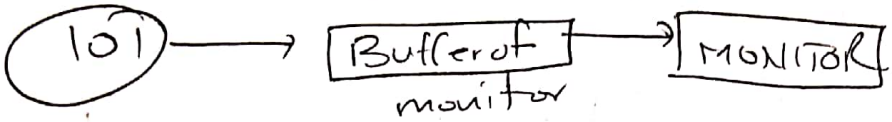
⊗ I/O devices have buffer associated with it.

Ex:

→ Calculator software (.exe format)



Result again placed in RAM



⊗ Similar Fx. of word doc.

Hard disk → i/p / o/p device

Keyboard → only i/p device

Monitor (Non-Touch) → only o/p device

HDD → i/p & o/p.

1) Exe 2) i/o 3) Waiting ⇒ Can happen in any order

Processes

Turn Around Time: Waiting Time + Burst time + I/O time

(HDD)

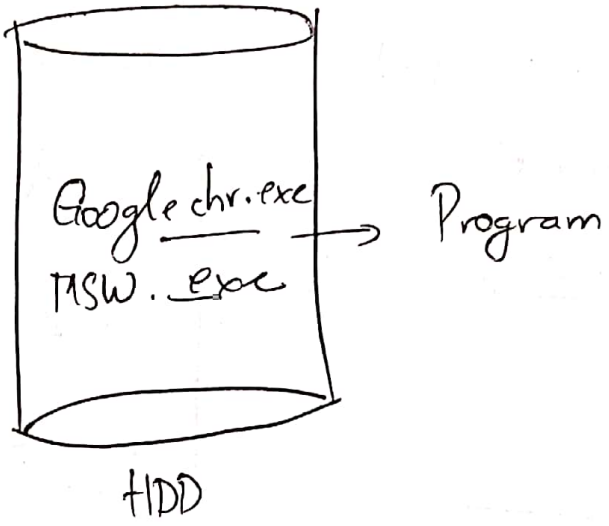
P_1 → RAM → Completed exe

(Program) 1AM 3AM

↳ Removed from Ram

Turn around time

Burst time → exe time



→ Chrome.exe → double clicked → CPU searches .exe in either RAM/HDD

→ New chrome.exe created in HDD → this is called a process

New Instance of gchrome in HDD

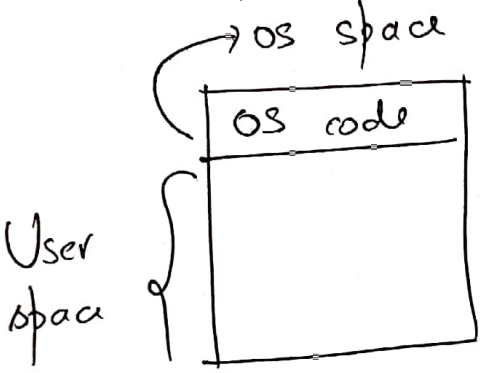
↳ copys of programs are called processes.

States of a process:

1) New state (when g.exe file is clicked)
2) Ready state (Process is waiting for exe or i/o)
3) Running state (Process is being exe by CPU)
4) I/o state (In RAM & I/o event) → also called blocked state
5) Terminated state ⇒ When process is terminate
6) Suspend Ready 7) Suspend wait → Remove

Degree of Multiprogramming:

of processes that can be placed in a RAM.



Assume:

Size of process ⇒ Same

$$\#P = \frac{\text{Size of Ram}}{\text{Size of 1 process}}$$

$$= \frac{2^{32}}{2^{12}} = \boxed{2^{20}}$$

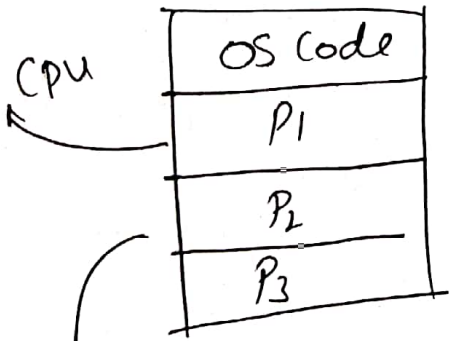
Degree of multiprogramming

2^{20} → Max # of processes which can be present in RAM of a comp

Types of OS

1) Batch OS → degree of MP = 1 (Only 1 process)

→ Usually more than 1 process in ram



Multiple things can happen

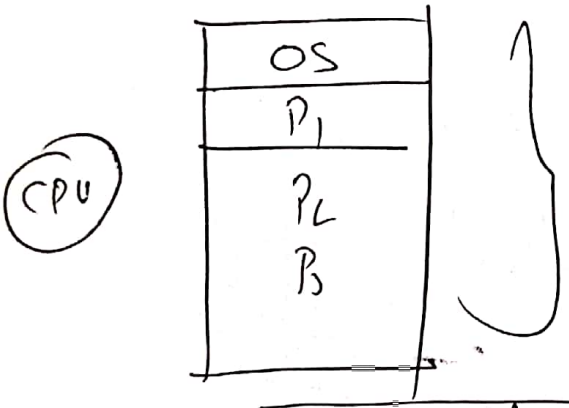
→ P_1 can be doing only one thing at a time

→ CPU idle time.

CPU efficiency : $\frac{\text{Useful time of CPU (Using)}}{\text{Total time of CPU}}$

② Multiprogramming OS:

CPU = 1



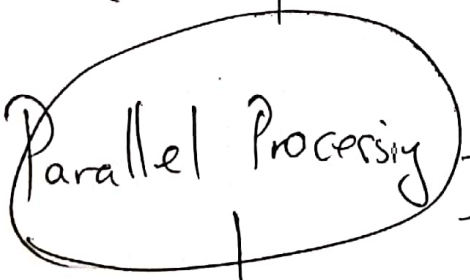
More than 1 process in the RAM.

→ CPU eff inc since CPU can do more things at once

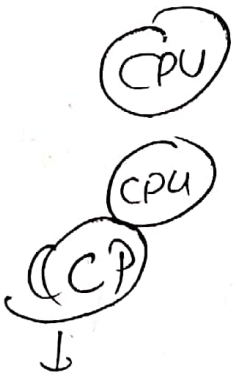
⇒ Concurrent processing

③ Multiprocessing OS

(Multiple CPUs)



faster than



Parallel processing

exe → more than one processing

↓

Cost Low.

↓

System cost ↑

# cores in	processor

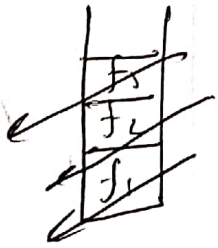
⊛ Examples → Multiprogramming OS

Passive Entity → Program
Active Entity → Process

Creates:

Program → f_1, f_2, $f_3 \rightarrow f_2$ } Nested fns

Function calls → In stack (according to calling)

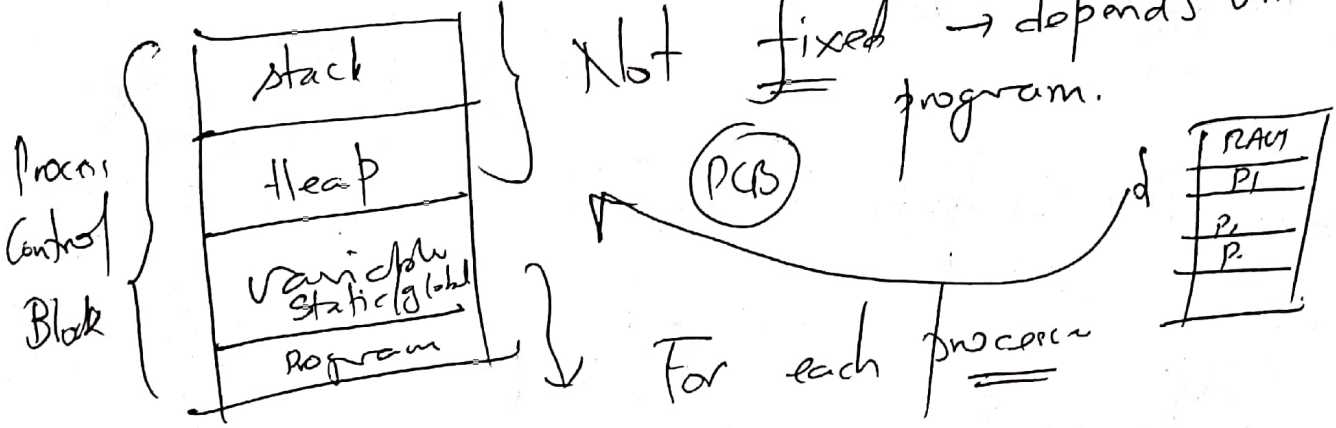


⊛ malloc() } Dynamic mem. allocation
calloc() }

Allocation in run time

↓

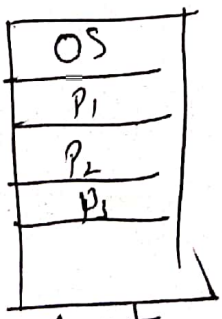
Created in Heap.



Attributes / Properties of a process

1) Process ID (unique number) ⇒ pid
2) Program Counter

Scheduler (Part of OS) → Code to select the process.



→ CPU

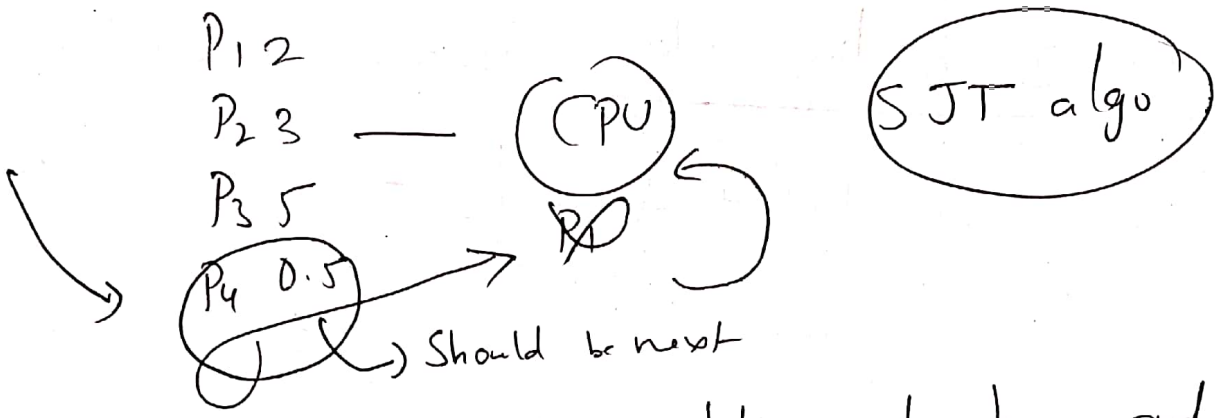
Done by part of

OS → has to select a process

FCFS → First Come First Serve → One such scheduling algos.

→ exe ⓞ using FCFS.

→ SJF → shortest job first algorithm
(according to time)
shortest exe time



P_3 must be abruptly stopped temp and then P_4 must be executed

Ⓧ If higher priority proces comes into RAM.
Low priority P → preempted.

P_3 → must be started processing again from the place where we left out

⇒ Program counter → Line # → Saves instruction # from where the process needs to be started again.

3) General Purpose Reg → Register values which are being used by the stopped process. → storing registers

→ List of open files → i/o processes. → P3

→ " " " devices →

Timestamp:

PCB of one process must not overlap each other

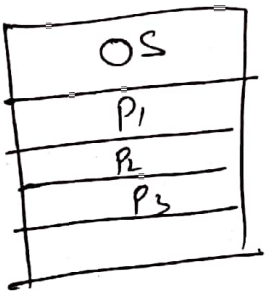
Protection

CPU Scheduling Algos: Process Control Block

Process → PCB + Attributes → Context of process

pid etc.
Lof

Long term scheduler →



Higher Priority → 1st exe ascending order

→ Short term scheduler (Part of OS code)
CRAM → CPU)

↳ Long term scheduler → HDD → RAM

⊗ ⊗ Assume RAM is full
⊗ Every process will have priority
Higher # priority → Earlier