Operating System

- System Software: operates and controls the computer system and provides a platform to run application software.

- Application Software: performs specific task for the user.

- Operating System: It is a piece of software that manages all the resources of a computer system, both hardware and software and provides an environment in which the user can execute his/her programs in an convenient and efficient manner by hiding underlying complexity of the hardware and acting as a resource manager.

- What if there is no operating System?
  - ⇒ Bulky and complex app.
  - ⇒ Resources explotiation by 1 app
  - ⇒ No Memory protection.

- What is an OS made up of?
  - ⇒ Collection of System Software.

- Functions of operating system:
  - ⇒ Access to the computer hardware.
  - ⇒ Interface between the user and hardware.
  - ⇒ Resource management {memory, device, file, security}
  - ⇒ Hides the underlying complexity of the hardware.
  - ⇒ Facilites execution of application programs by providing isolation and protection.

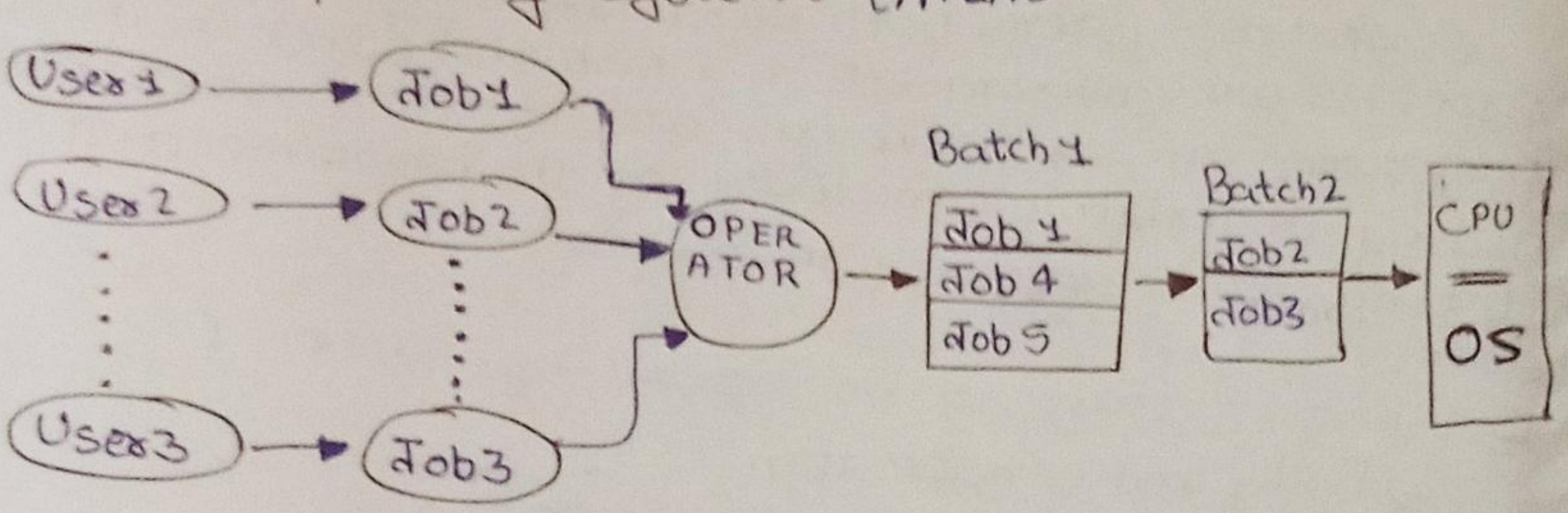
□ Goals of Operating System :-

i) Maximum CPU Utilisation
ii) Less process starvation
iii) Higher priority job execution

□ Types of Operating System :-

i) Single Process OS {MS-DOS, 1981}

ii) Batch Operating System : {ATLAS

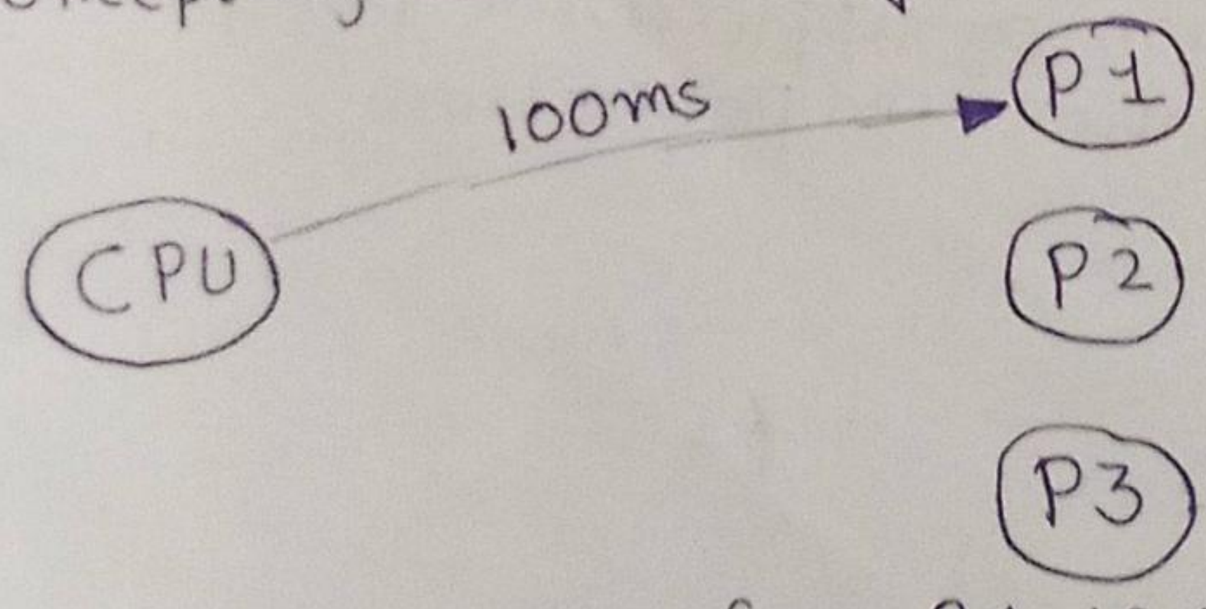


iii) Multiprogramming Operating System : It increases {THE} CPU utilisation by keeping multiple jobs in the memory, so that CPU always has one task to execute

⇒ Concept of context switching

PCB ⇒ Process Control Board

iv) Multitasking Operating System : Single CPU, context {CTSS} switching with a concept of Time sharing.



CPU will wait 100ms for P1, if it does not complete CPU moves to P2.

⑤ Multiprocessing Operating System: In this type, we can have multiple CPU {Windows}

In all other OS other than Multiprocessing OS, if CPU fails in all the jobs will not be completed.

⑥ Distributed Operating System : {Loosely coupled}

In this all their are different CPU, machines all are connected with a single operating system with the help of network.

⑦ Real-Time Operating System : This OS is used when the very error free process like in rocket launch.

Multitasking Vs Multithreading

Process => Program under execution

Thread is like small weight process

Multitasking

- Work on the concept of more than 1 process
- Process scheduling
- No. of CPU 1.
- Isolation and memory protection exists

Multithreading

- Work on the concept of more than 1 threads.
- Thread Schdeuling
- No. of CPU >= 1
- No isolation and memory protection.

☐ Thread Schdeuling: Threads are scheduled for execution based on their priority. Even though threads are executing within the runtime, all threads are assigned processor time slices by the operating system

☐ Difference between Thread Context Switching And Process Context Switching :

Thread Context Switching	Process Context Switching
⊙ OS saves current state and switches to another thread of same process	⊙ OS saves current state of process and switches to another process by restoring its state
⊙ Doesn't includes switching of memory address space	⊙ Includes switching of memory address space
⊙ Fast switching	⊙ Slow switching
⊙ CPU's cache is preserved	⊙ CPU's cache state is flushed.

Components Of Operating System

1. Kernel: It is that part of operating system which interacts directly with the hardware and performs the most crucial tasks
   i) Heart of OS/Core component
   ii) Very first part of OS to load on start-up

2: User Space: Where application software runs apps don't have priviliged access to the underlying hardware. It interects with kernel
   i) GUI
   ii) CLI

☐ A shell, also known as command interpreter, is that part of the operating system that receives commands from the users and get them executed.

☐ Functions of Kernel:

→ Process Management:
a) Scheduling process and threads of the CPU.
b) Suspending and Resuming process.
c) Providing mechanism for system process synchronization or process communication

→ Memory Management:
a) Allocating and Deallocating memory as per need.
b) Keeping track which part of memory is used and by which process.

→ File Management:
a) Creating and deleting file
b) Creating and deleting directories to organize file
c) Mapping Files in secondary storage.
d) backup support onto a stable storage media.

→ I/O Management:
a) manage and control all I/O devices.
   i) Spooling
   ii) Buffering
   iii) Caching

☐ Types of Kernels:

→ Monolithic Kernel:
a) Oldest Kernel
b) All Functions (Process, Memory, File, I/O management) present in single kernel.
c) Bulky in size
d) High performance as communication is fast.
e) Less Reliable, one module crashes → whole system/ Kernel is down.
f) Memory required is high.

→ Micro Kernel :

a) Only important process management and memory management are in kernel.

b) File management and I/O management are in user-space

adv c) Less bulky

adv d) More Reliable

adv e) More Stable

dis f) Performance is slow

dis g) Overhead switching b/w user mode & kernel mode

→ Hybrid Kernel :

a) Advantages of both worlds (File mgmt. in user space and rest in Kernel space.)

b) Combined approach

c) Speed and design of monolithic

d) Modularity and stability of Microlithic

Note: How will communication happen between user mode and Kernel mode

⇒ They communicate ~~between~~ through IPC {Inter Process Communication}

⊙ Two processes executing independently, having independent memory space (Memory protection), But some may need to communicate to work

⊙ Done by shared memory and message passing.

⊙ Message Passing

Apps interact with Kernel using System Call

→ System calls are implemented in C.

→ Transition from user space to Kernel space is done by software interrupt.

□ System Call : A system call is the mechanism using which a user program can request a service from Kernel for which it does not have permession to perform.
User program typically do not have the permission to perform operations like I/O devices and communicating other programs.
System calls are the only way through which a process can go into Kernel mode from user mode.

□ Types of system Calls
i) Process control (end, abort, load, execute)
ii) File Management (create file, delete file, open)
iii) Device Management (request device, release device, read)
iv) Information maintainance (get process, set process)
v) Communication Management (create, send, recieve message)

Example of Windows And Unix System Calls

Category	Window	Unix
Process Control	CreateProcess() Exit Process() WaitForSingleObject()	fork() exit() wait()
File Managment	CreateFile() Read File() Write File() Set Filesecurity()	Open() read() write() chmod()

What happens when you turn your computer

i) Power On

ii) CPU intialize itself and looks for a firmware program BIOS (Basic Input Output System) stored in BIOS chip.

iii) CPU runs BIOS/UEFI which test and intialize system hardware. BIOS load configration settings If something is not appropiate error is thrown and boot process is stopped.
This is called **POST** (Power On Self Test) Process.

(iv) BIOS/UEFI will handoff responsbility for booting your PC to your OS's bootloader

→ Bootdevice : It is a device/area, where the instructions related to OS's boot is written. It can be Disk (HDD SSD, CD)

→ Bootloader : It is a program, that when executes, runs the OS.

→ MBR → Master Boot Record. (stored in 0th index of Memory. Used by BIOS supported system

→ EFI : Partiation in disk (Bootloader inside this partition). Used by UEFI

(v) The bootloader is a small program that has the large task of booting the rest of operating system.

Windows uses Window Boot Manager

Linux uses GRUB (Grand Unified Bootloader)

MAC uses boot.efi

32-bit and 64-bit operating System

- 32-bit OS: It has 32-bit registes, and it can access 2^32 unique memory address. i.e 4GB of RAM. It can process 32 bits of data and information.
- 64-bit OS: It has 64-bit registes, and it can access 2^64 unique memory address. It can have 64 bits of data and information.

Types Of Storages

- Registes: Smallest unit of storage. It is past of CPU memory itself.
- Cache: Additional memory system that temporaily stores frequently used instruction and data for quicker processing by CPU.
- Main Memory: RAM
- Secondary Storage: Storage media, on which compoter can store data and programs.

★ Comparison:

- Cost: Primary storage are more costly
  Registers are most expensive due to expensive semiconduct
  Secondary storage are cheaper than primary.
- Access Speed: Primary has higher access speed than secondary
  ⊙ Register has highest access speed, then came cache, then memory.
- Volatility: Primary memory is volatile
  Secondary memory is non volatile
- Storage Size: Secondary has more space.

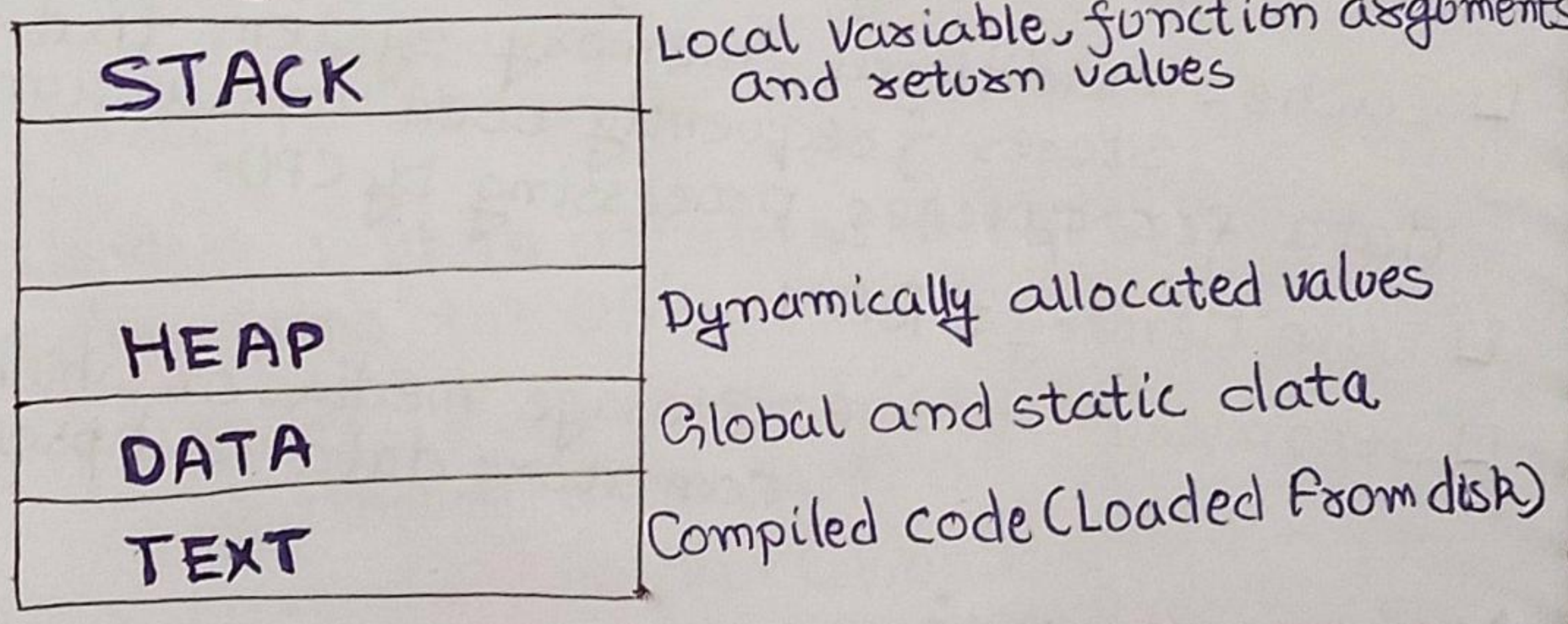
Introduction To Process

→ Program : Compiled code, ready to execute.
→ Process : Program under execution
→ How OS creates a process : Converting program into process.

STEPS :

a) Load the program and static data into memory.
b) Allocate runtime stack
c) Heap memory allocation
d) IO tasks
e) OS handoffs control to main()

☐ Architecture of Process



☐ Attributes Of Process :

a) Features that allows identifying a process uniquely
b) Process table
   ⊙ All process are being tracked by OS using a table like data structure
   ⊙ Each entry in the table is Process control block (PCB).
c) PCB : stores info/attributes of a process.
   ⊙ Data structure used for each process, that stores information of a process such as process id, program counter, process state, priority etc.

☐ PCB Structure:

Process Id	{Unique identifier}
Program Counter	{Next inst. address of a program}
Process State	{Stores process state}
Priority	{Based on priority a process get CPU time}
Register	
List of open files	
List of open devices	

☐ Register in the PCB: It is a data structure. When a process is running and it's time slice expires, the current value of process specific register would be stored in PCB and the process would be swapped out. When the process is scheduled to be run, the register value is read from the PCB and written to the CPU register.

Different Process State In Operating System

☐ Process State: As Process states, it changes state. Each process may be in one of the following states.

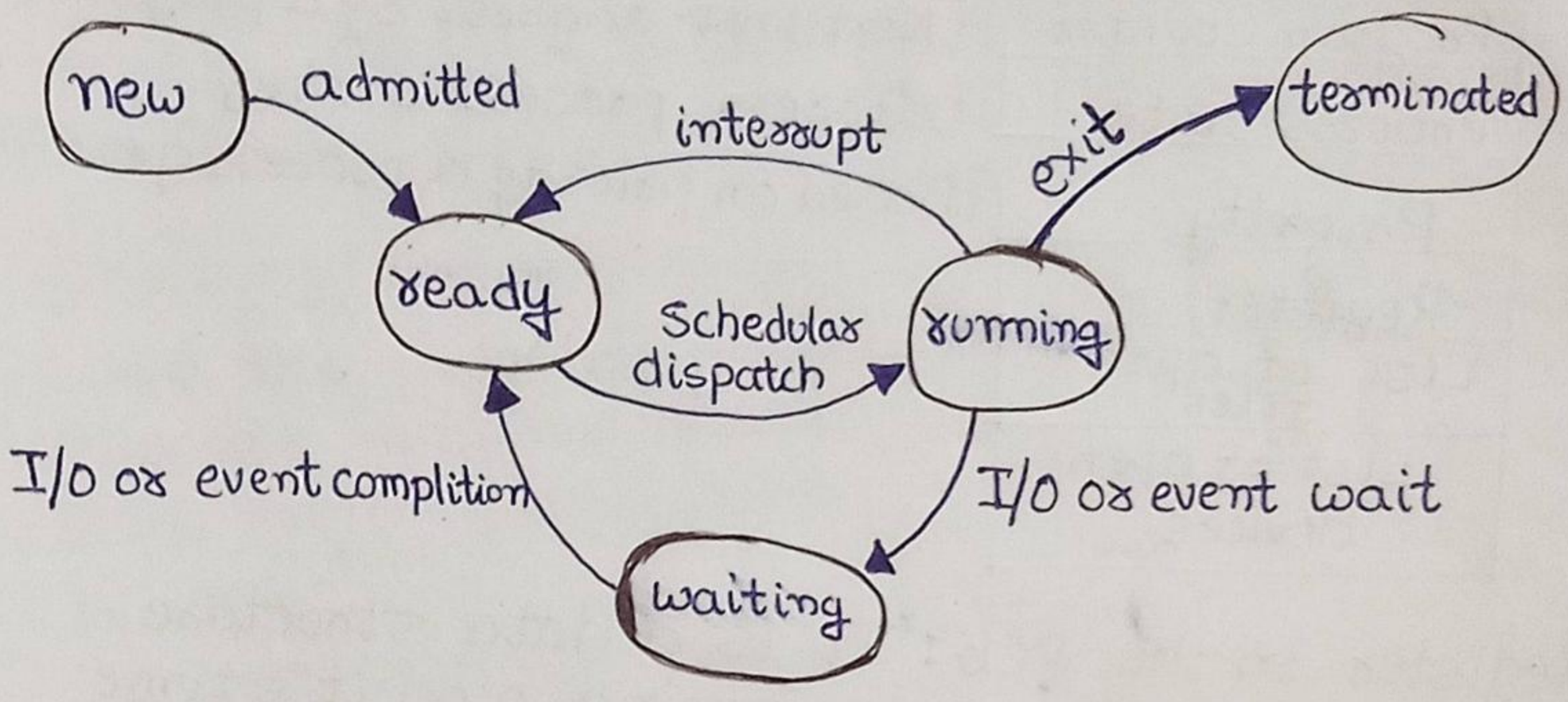
a) **New**: OS is about to pick the program and convert it into process or the process is being created.

b) **Run**: Instructions are being executed, CPU is allocate

c) **Waiting**: Waiting for IO (Input/Output)

d) **Ready**: The process is in memory, waiting to be assigned to a processor.

e) **Terminated**: The process has finished execution. PCB entry removed from process table.



- Process Queues:

a) Job Queue :
- → Processes is in new state.
- → Present in secondary memory.
- → Job Schedulax (Long term schedulax (LTS)) picks from the pool and loads them into memory for execution processes

b) Ready Queue:
- → Processes is in ready state
- → Present in main memory.
- → CPU Schedulax (Short Term Schedulax (STS)) picks process from ready queue and dispatch it to CPU.

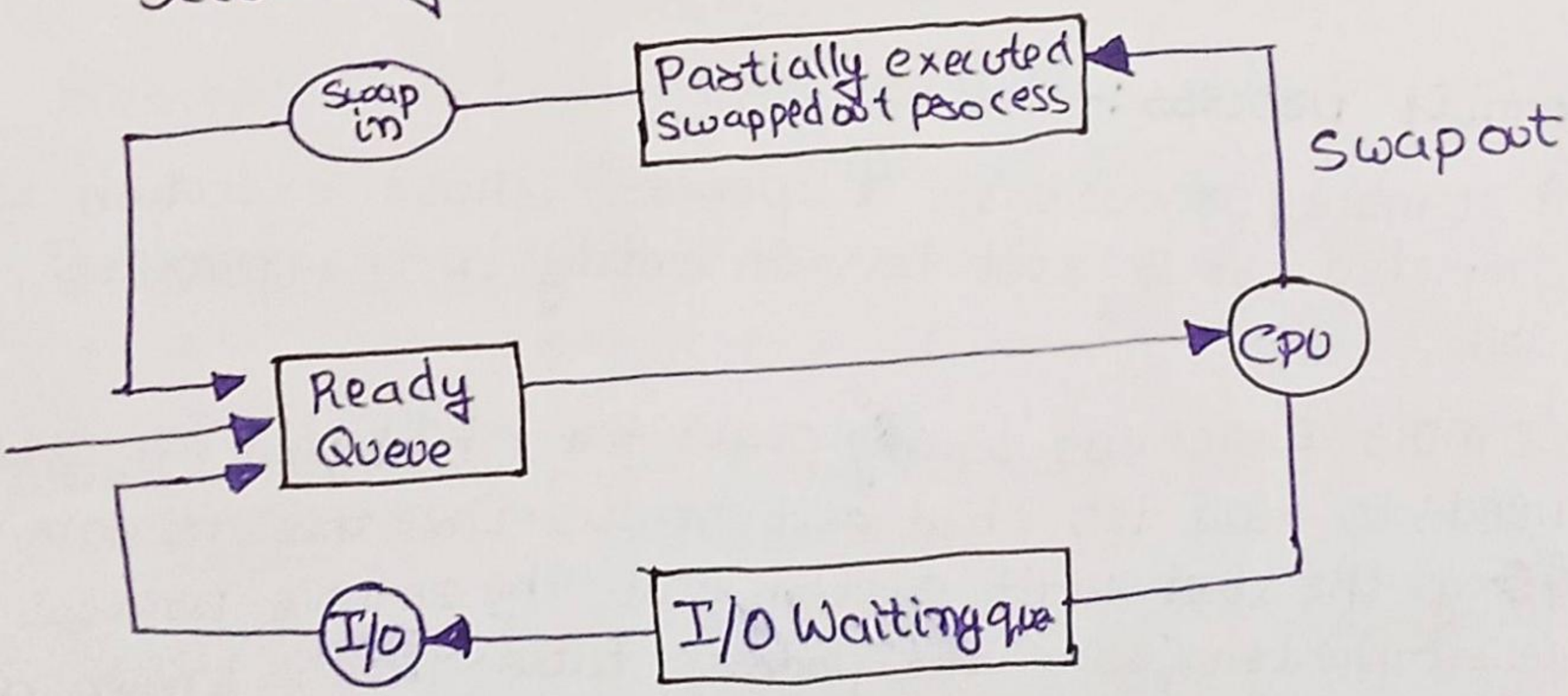
c) Waiting Queue:
- → Processes in wait state

- Degree of multiprogramming : The number of processes in the memory.
  - → LTS control degree of moltiprogramming.
- Dispatachex : The module of the OS that gives contrd of CPU to a process detected / selected by STS.

□ MTS : Medium Term Schedular

★ Swapping :-

→ Time-sharing system may have MTS
→ Remove processes from memory to reduce degree of multiprogramming.
→ These removed processes can be reintroduced into memory, and its execution can be continued where we left off. This is called swapping.
→ Swap-in or swap out is done by MTS.
→ Swapping is the mechanism in which process can be swapped temporarily out of main memory to secondary storage and make the memory available to other processes. At some later time, the system swap back the process from the secondary storage.



Context Switching :-

→ Switching the CPU to another process requires performing a state save of the current process and a state restore of a different process.

→ When this occurs, the kernel saves the context of the old process in its PCB and loads the saved context of the new process scheduled to run.

→ It is pure overhead, because the system does no useful work while switching.

Orphan process :-

→ The process whose parent process has been terminated and is still running.

→ Orphan processes are adopted by init process.

→ Init is the first process of OS (Linux).

Zombie process :-

→ A zombie process is a process whose execution is completed but it still has an entity in the process table.

→ Zombie processes usually occur for child process still needs to read its child exit status. Once this is done using the wait system call, the zombie process is eliminated from the process table. This is known as reaping the zombie process.

→ It is because parent process may call wait() on child process for a longer time duration and child process may get terminated much earlier.

Scheduling Algorithms :- It is a way ~~to~~ of selecting a process from ready queue and put it into CPU.

i) Pre-emtive : Process may execute in parts

ii) Non Pre-emtive : Fully executed in one time

☐ Pre-emtive Scheduling Algorithm
- → SRTF {Shortest Remaining Task First} **
- → LRTF {Longest Remaining Task First}
- → Round Robbin **
- → Priority Based

☐ Non Pre-emitive Scheduling Algorithm
- → FCFS {First Come First Serve}
- → SJF {Shortest Job First}
- → LJF {Longest Job First}
- → HRRN {Highest Response Ratio Next}
- → Multilevel Queue

☐ Arrival Time :- The Time at which process enter the ready queue or state.

☐ Burst Time :- It is the time required by a process to get execute on CPU. (Duration)

☐ Completion Time :- The Time at which process complete its execution.

→ Turn Around Time :- Completion Time - Arrival Time

→ Waiting Time :- Turn Around Time - Burst Time

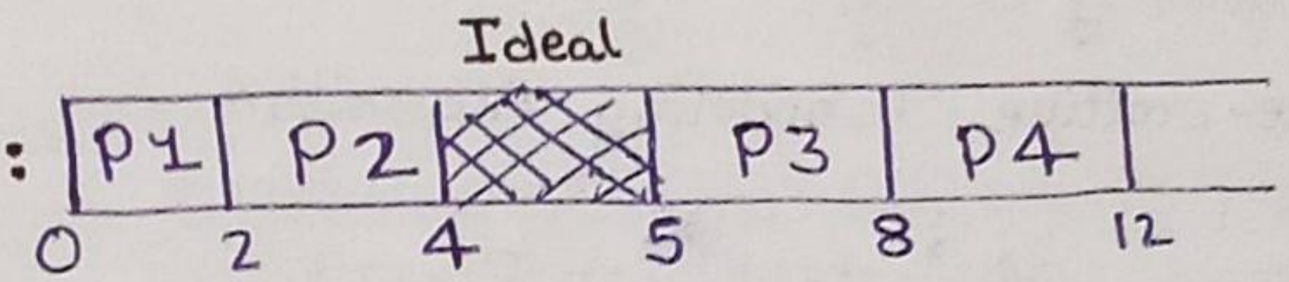
→ Response Time :- [(The Time at which a process gets CPU first Time) - (Arrival Time)]

☐ First Come First Serve (FCFS)

Criteria : Arrival Time
Mode : Non-Premetive

Process No	Arrival Time	Burst Time	Completion Time	TAT	WT	RT
✓ P1	0	2	2	2	0	0
✓ P2	1	2	4	3	1	1
✓ P3	5	3	8	3	0	0
✓ P4	6	4	12	6	2	2

Grant Chart:



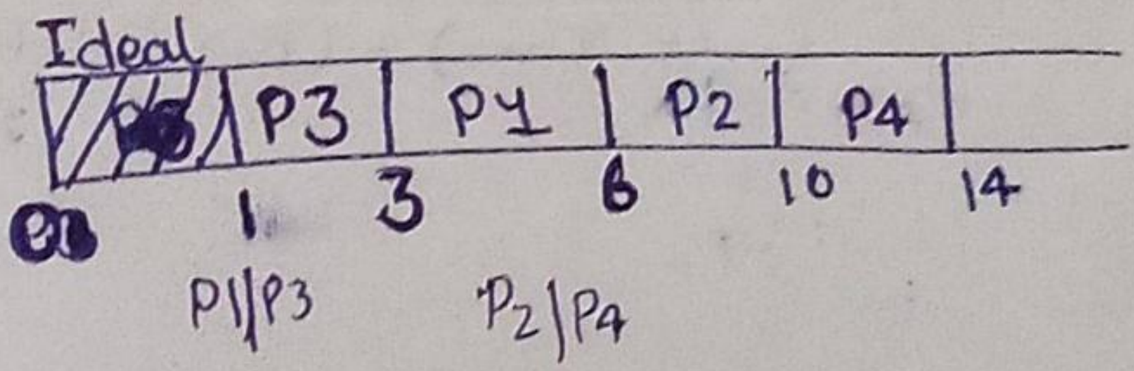
→ In case of Non-Premetive Response Time will be equal to Waiting Time.

☐ Shortest Job First (SJF)

Criteria: Burst Time
Modes: Non-Preemetive

Process Id	Arrival Time	Burst Time	Completion Time	TAT	WT	RT
P1	1	3	6	5	2	2
P2	2	4	10	8	4	4
✓ P3	1	2	3	2	0	0
P4	4	4	14	10	6	6

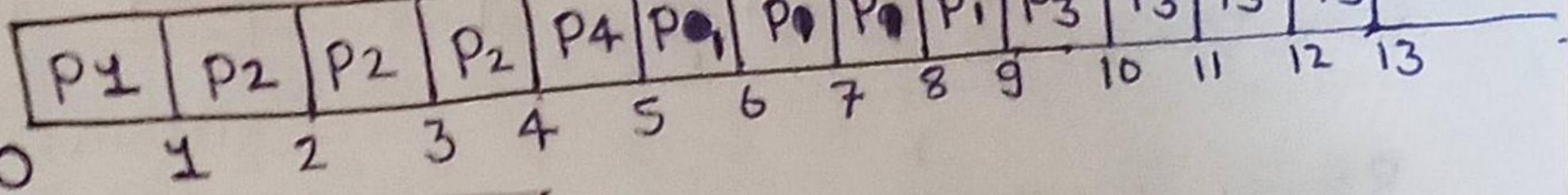
Grant Chart:



☐ Shortest Remaining Task First

Process Id	Arrival Time	Burst Time	Completion Time	TAT	WT	RT
P1	0	~~5~~ 4	9	9	4	0
P2	1	32	4	3	0	0
P3	2	4	13	11	7	7
P4	4	1	5	1	0	0

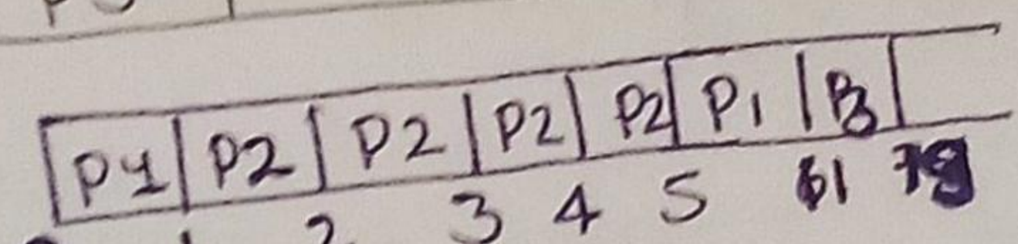
Gixant Chart:

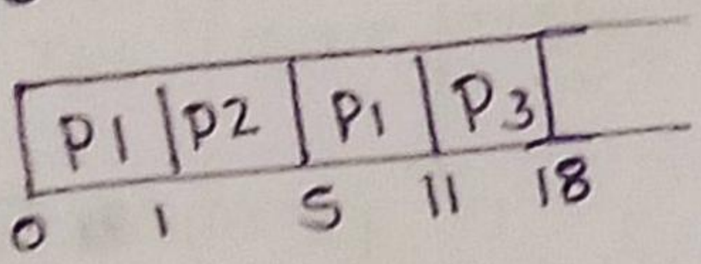


Qus 1)

PID	Arrival	Burst
P1	0	~~7~~ 6
✓P2	1	~~4~~ 3
P3	2	8

Gixant Chart





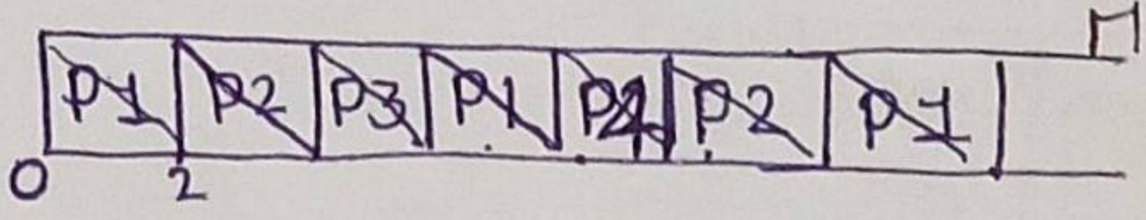
☐ Round Robbin Scheduling Algorithm :-

Process Id	Arrival Time	Burst Time	Complition Time	TAT	WT	RT
✓ P1	0	$5_{3,1}$	12	12	7	0
✓ P2	1	4_2	11	10	6	1
✓ P3	2	2	6	4	2	2
✓ P4	4	1	9	5	4	4

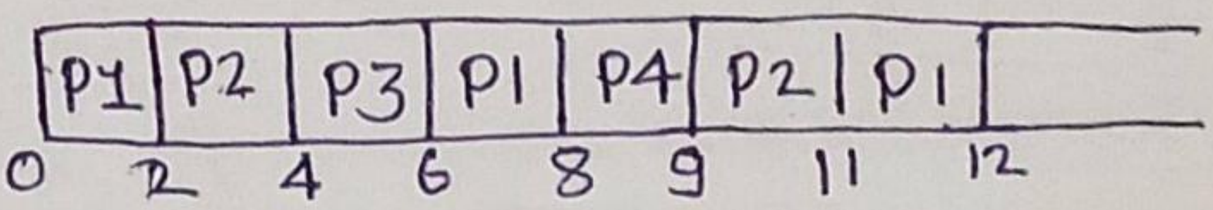
given TQ = 2;

Criteria: Time Quantum
Mode: Pre-emtive

Ready Queue



Running Queue

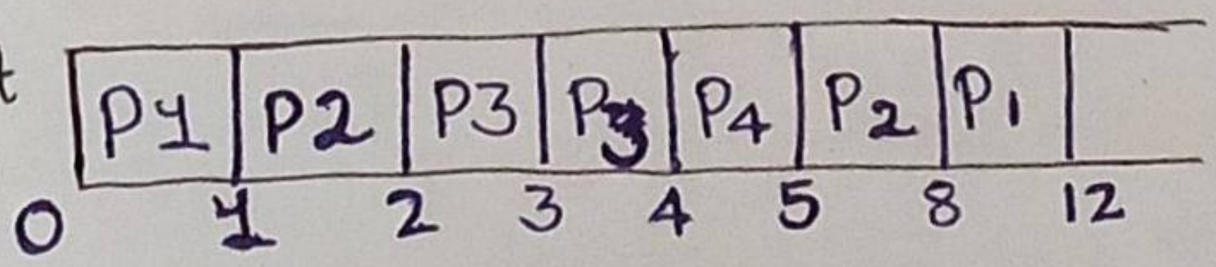


☐ Priority Scheduling :-

Priority	Process Id	Arrival Time	Burst Time	Complition Time	TAT	WT	RT
10	P1	0	5_4	12	12	7	
20	P2	1	4_3	8	7	3	
30	P3	2	$2_{1,0}$	4	2	0	
40	P4	4	1_0	5	1	0	

Criteria: Priority
Mode: Preemtive

Gnant Chart

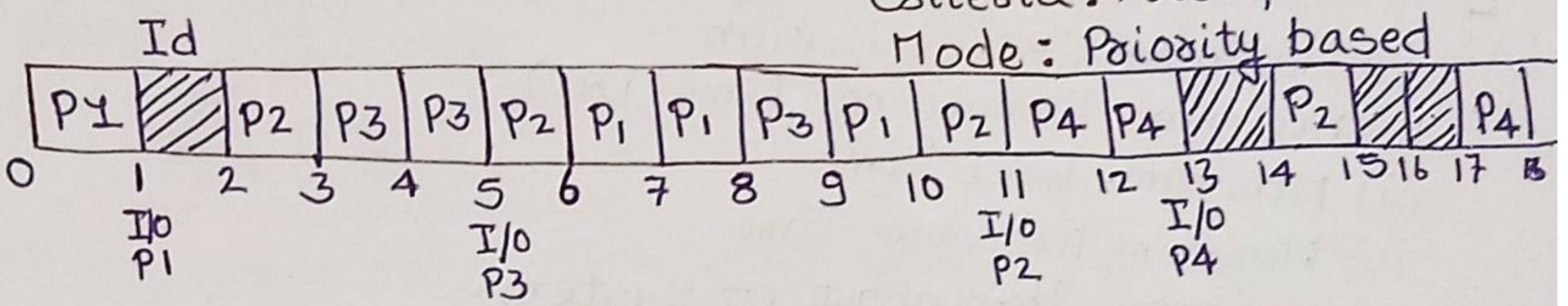


☐ Mix Burst-Time Scheduling :-

{Lowest no. higher priority}

Process Id	Arrival Time	Priority	CPU	I/O	CPU	Complition Time
✓ P1	0	2	1_0	5	3	10
✓ P2	2	3	3_{21}	3	1	15
✓ P3	3	1	2_{10}	3	1	9
✓ P4	3	4	2	4	1	18

Criteria: Preemptive
Mode: Priority based



Theory Of Process Scheduling

☐ Process Scheduling :- Many processes are kept in the memory at a time, when a process must wait or time quantum expires, the OS takes the CPU away from that process and gives the CPU to another process and this pattern continues.

☐ CPU scheduler :- Whenever the CPU became ideal, OS must select one process from the ready queue to be executed (Done by STS).

☐ Non-Preemptive Scheduler :-

⊙ Once CPU has been allocated to a process, the process keeps the CPU until it releases CPU either by terminating or by switching to wait state.

⊙ Starvation, as a process with long burst time may starve less burst time processes.

⊙ Low CPU utilisation

□ Preemptive scheduling :-

⊙ CPU is taken away from a process after quantum time expires along with terminating or switching to wait-state.

⊙ Less starvation.

⊙ High CPU utilisation

□ Goals of CPU scheduling :-

i) Maximum CPU utilisation
ii) Minimum Turnaround Time (TAT)
iii) Minimum Wait Time
iv) Minimum Response Time
v) Maximum Throughput of system.

□ Throughput :- No. of processes completed per unit time.

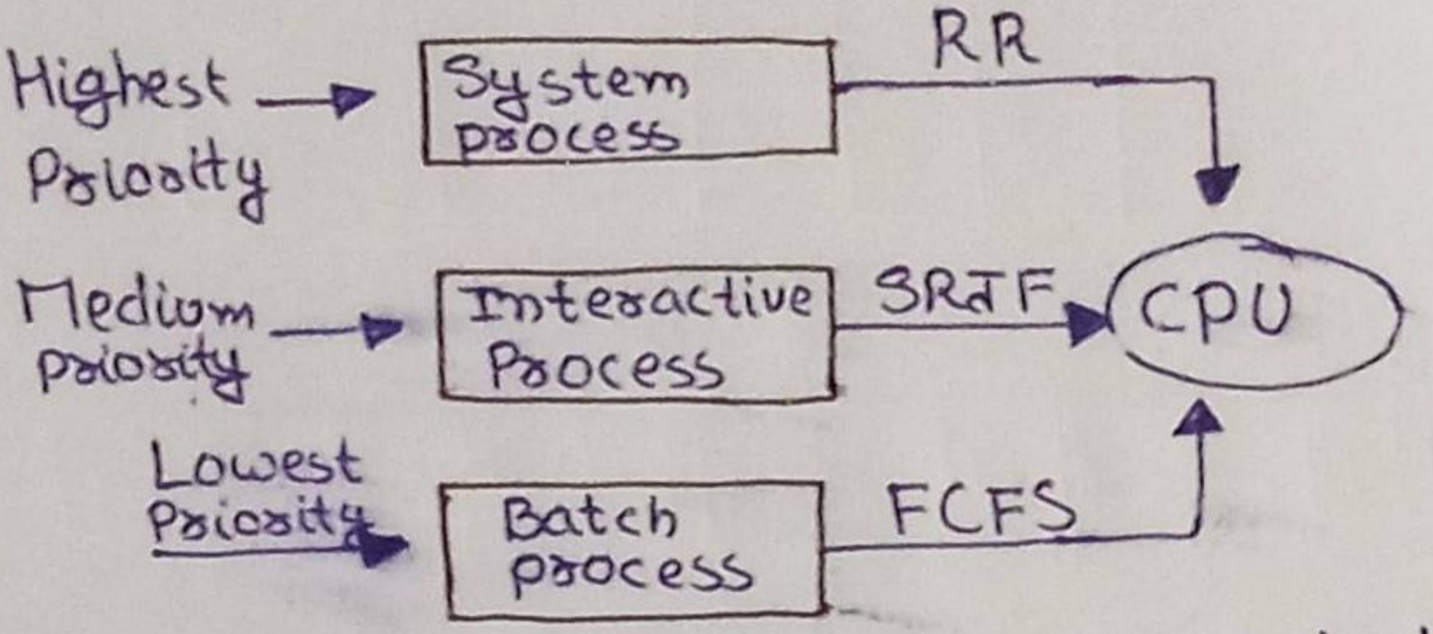
□ FCFS {First Come First Serve}

⊙ Whichever process comes first in the ready queue will be given CPU first.

⊙ In this, if one process has longer Burst Time - It will have mayor effect on average Waiting Time of different process.

→ Convoy Effect is a situation where many processes, who need to use a resource for a short time, are blocked by one process holding that resource for a long time.
This cause poor resource management.

□ Multi level Scheduling : There must be different queues for every process. When a system process will come, we'll place it in the queue interactive process in other queue and batch process in other one.



→ Every queue have their own scheduling algorithm

Advantage :

i) Categorizing the processes according to their different life scenario.

1

Disadvantage :

i) Process Starvation for Medium and Low priority

□ Multi Level Feedback Queue :