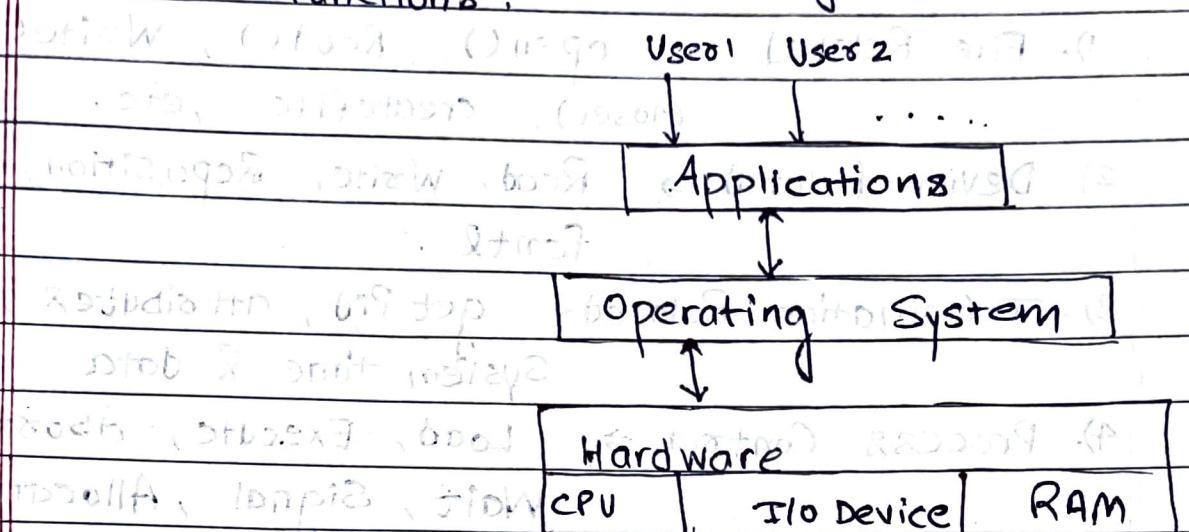


# Introduction to Operating System and its functions.



Convenience.

Throughput

Functions :-

- 1). Resource Management
- 2). Process Management (CPU Scheduling)
- 3). Storage Management (HD)
- 4). Memory Management (RAM)
- 5). Security

Multiprogrammed OS (Non preemptive)

↳ IDLENESS

↳ Execute first process then only switch to second completely

Multitasking OS / Time Sharing OS

↳ Responsiveness

Execute task on the basis of slots.

## "System Call"

- 1). File Related  $\Rightarrow$  open(), Read(), Write(), Close(), creatfile ,etc.
- 2). Device Related  $\Rightarrow$  Read, Write, Reposition, ioctl, fcntl .
- 3). Information Related  $\Rightarrow$  get Pid , attributes , get System time & data
- 4). Process Control  $\Rightarrow$  Load, Execute, Abort, Fork, Join, Wait, Signal , Allocate, etc
- 5). Communication = Pipe() , Create / delete Conn Shmget()

## "Fork"

Fork System Call is used for creating new process , which is called child process , which runs concurrently with the process that made the fork() call.

A child process uses same PC (program Counter) same CPU registers ,

(switching next 20 bits of program counter)

fork()

(return type : int)

parent



20 parent shift 20 child shift +1

of 8 unsuccessful Returned to newly created child process OR caller

P (parent)

↓ fork

o C<sub>1</sub> ← +ve

fork

o C<sub>2</sub> ← -ve

fork

o C<sub>3</sub> ← +ve

fork

o C<sub>4</sub> ← -ve

fork

Question 1 On Fork

Q.1. int main()

{ return 0;

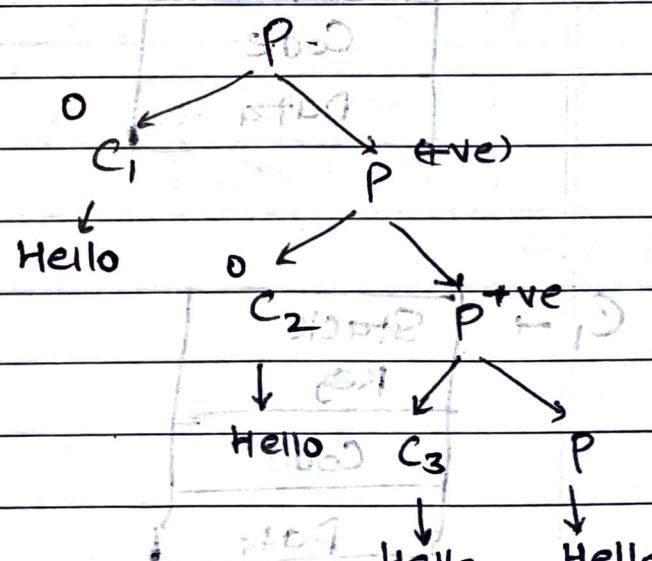
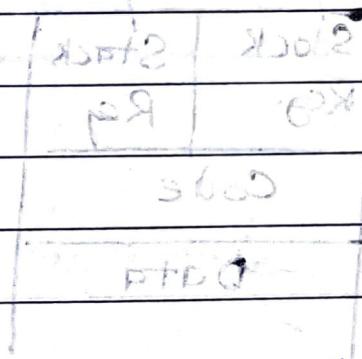
base if a pair if (fork() && fork())

base write if (fork());

print("Hello\n");

return 0;

}



Sol → Hello (4 times)

# Process vs Threads

User level

## Process

System calls involved in the process

OS treat different process differently

Different process have diff. copies of data, file codes.

Context switching is slower

Blocking a process will not block another

Interdep Independent

## Threads

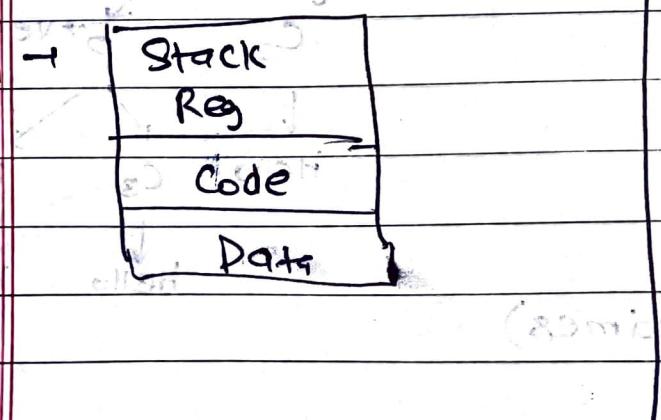
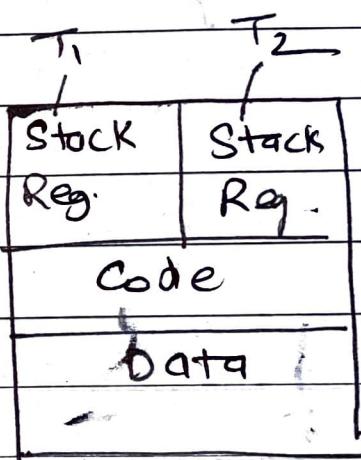
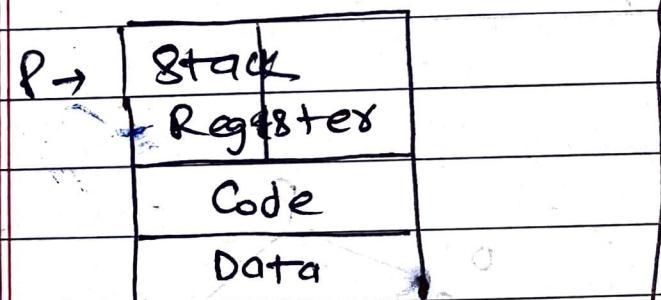
i) There is no system calls involved

All user level threads treated as single task for OS  
Threads share same copy of data and fixe code

Context switching is faster

Blocking a thread will block entire process

Interdependent



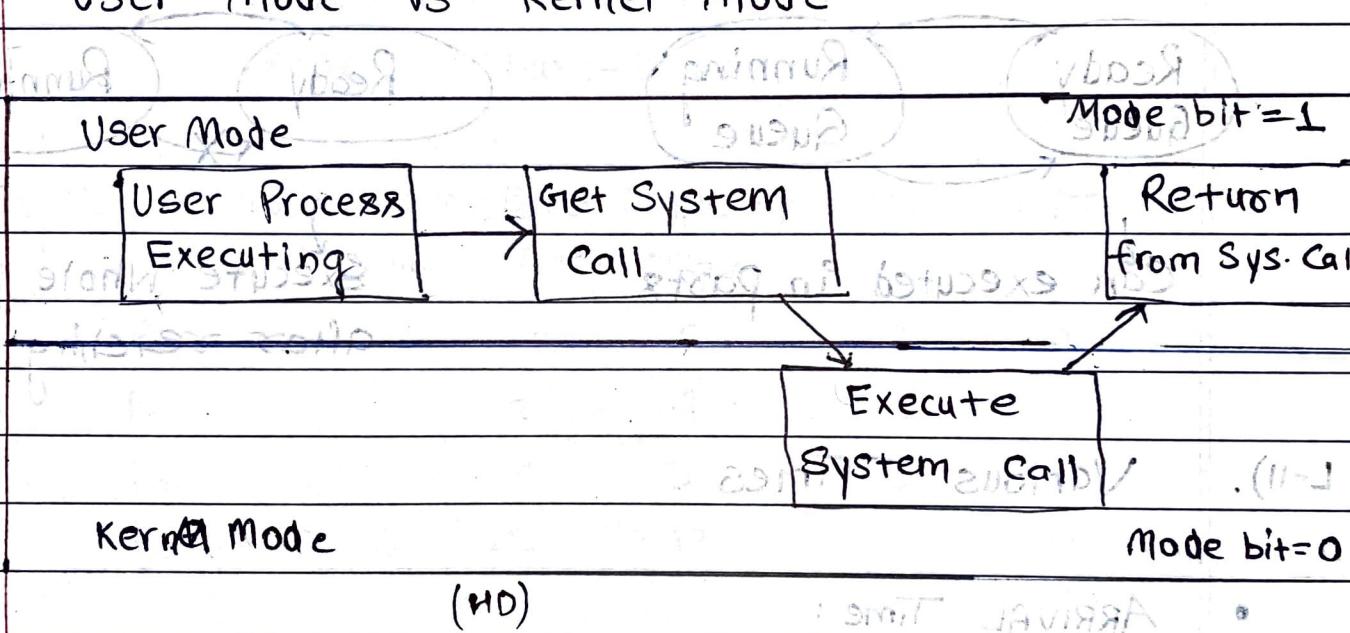
## User level Thread

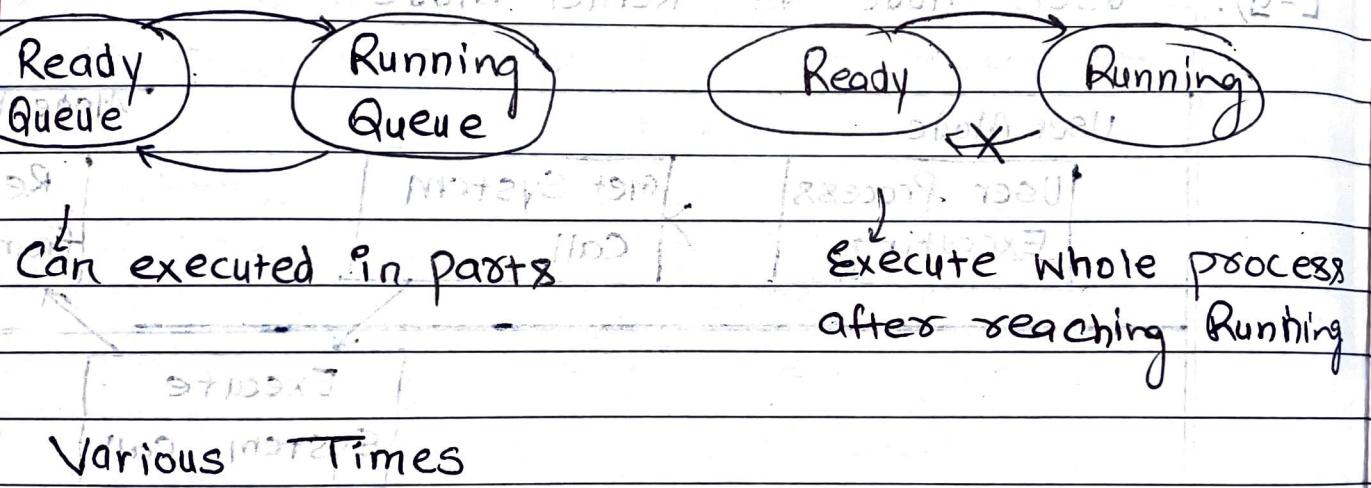
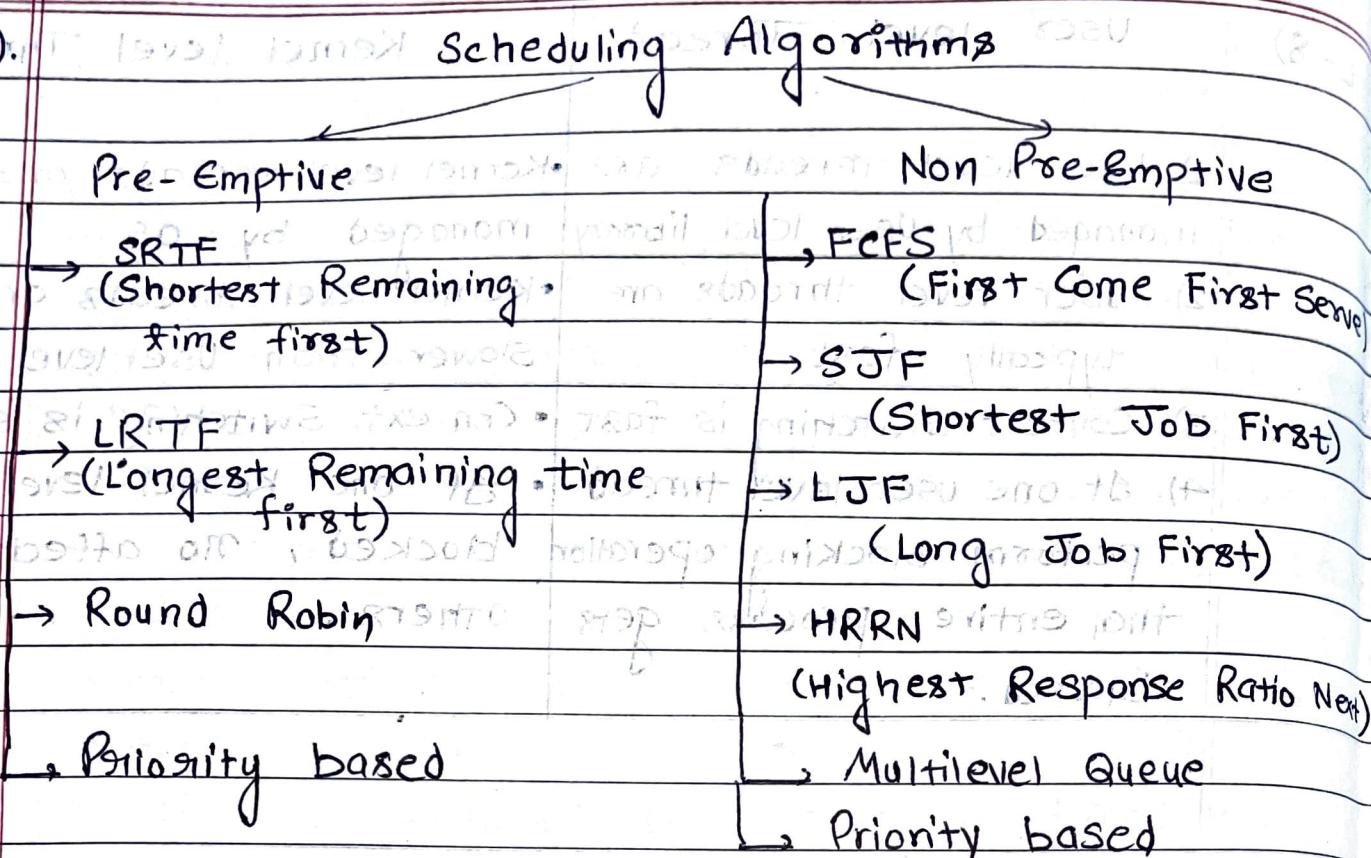
- 1). User level threads are managed by user level library.
- 2). User level threads are typically fast.
- 3). Context Switching is fast.
- 4). If one user level thread performs blocking operation then entire process gets blocked.

## Kernel level Thread

- Kernel level threads are managed by OS.
- Kernel level threads are slower than User level.
- Context Switching is slow.
- If one kernel level thread blocked, no affect on others.

## User Mode Vs Kernel Mode





ARRIVAL Time :

The time at which process enter the ready Queue.

BURST Time : Time Required by a process to get executed on CPU.

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 : {Time at which a process get CPU first time - Arrival time}

Ex :  $\text{Arrival Time} = 0$   
 $\text{Burst Time} = 3$   
 $\text{Completion Time} = 3$

CPU Scheduling Algorithm →

- First Come First Serve (FCFS)

Criteria : "Arrival Time"

Mode : "Non-Preemptive"

Process No	Arrival Time	Burst Time	Completion Time	TAT	WT	RT
P <sub>1</sub>	0	2	2	2	0	0
P <sub>2</sub>	1	3	4	3	1	1
P <sub>3</sub>	5	3	8	3	0	0
P <sub>4</sub>	6	4	12	6	2	2

Grant Chart	P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>	P <sub>4</sub>
0	1	2	4	5
1	1	2	3	4
2	1	2	3	5
3	1	2	3	6
4	1	2	3	7
5	1	2	3	8
6	1	2	3	9
7	1	2	3	10
8	1	2	3	11
9	1	2	3	12

$$\text{Avg. Waiting Time} = \frac{3}{4}$$

# Shortest Job First (SJF) Algorithm

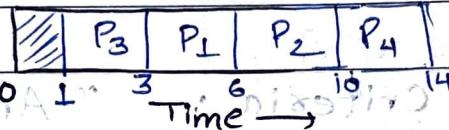
Criteria: "Burst Time"

Mode : Non-Preemptive

SJF Round Robin = SJF Burst by Circuit

Process	Arrival Time	Burst Time	Completion Time	TAT	WT	RT
No	Time	Time	Time			
P <sub>1</sub>	1	3	6	5	2	2
P <sub>2</sub>	2	4	10	8	4	4
P <sub>3</sub>	1	2	3	2	0	0
P <sub>4</sub>	4	4	14	10	6	6

Grant Chart



"Scheduling - non" : SJF

P<sub>1</sub>, P<sub>3</sub> → both arrived at 1

Now TAT = Burst time of P<sub>3</sub> + W<sub>1</sub> Smaller So P<sub>3</sub> will be selected.

Now P<sub>1</sub>, P<sub>2</sub> will be in the ready queue at 3.

Burst time (P<sub>1</sub>) < Burst Time (P<sub>2</sub>)

Now P<sub>2</sub>, P<sub>4</sub> → Burst time Same

So select on the basis of arrival time

In Non-preemptive Response time == WT

Average Turn Around Time

$$= \frac{5+8+2+10}{4}$$

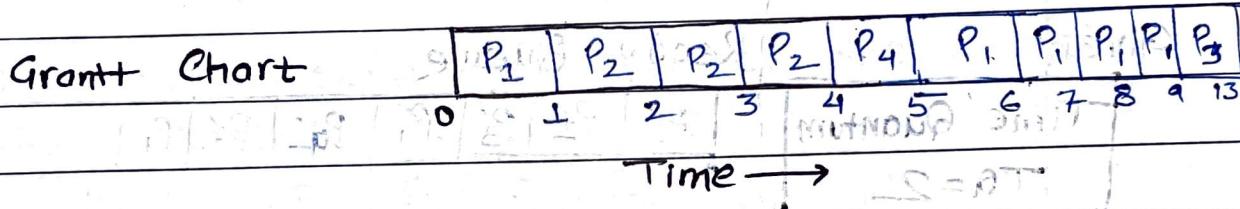
$$= 6.25$$

$$\text{Avg WT} = 3$$

# Shortest Remaining Time First (SRTF)

Criteria: "Burst Time" is Mode "Preemptive"

Process No	Arrival Time	Burst Time	Completion Time		TAT	WT	RT
			Initial	Final			
P <sub>1</sub>	0	5	5	9	9	9	0
P <sub>2</sub>	1	3	4	7	3	0	0
P <sub>3</sub>	2	4	13	17	11	5	7
P <sub>4</sub>	4	1	5	6	1	0	0



At 0 → P<sub>1</sub>(4)

At 1 → P<sub>1</sub>, P<sub>2</sub> (4, 3)

At 2 → P<sub>1</sub>, P<sub>2</sub>, P<sub>3</sub> (4, 2, 4)

At 3 → P<sub>1</sub>, P<sub>2</sub>, P<sub>3</sub> (4, 1, 4)

At 4 → P<sub>1</sub>, P<sub>2</sub>, P<sub>3</sub>, P<sub>4</sub> (4, 4, 1)

At 5 → P<sub>1</sub>, P<sub>3</sub> (4, 4) → Now Arrival

$$\text{Avg. Waiting Time} = \frac{24}{4} = 6$$

$$\text{Avg. Response Time} = \frac{7}{4} = 1.75$$

# • Round Robin (RR) CPU Scheduling Algorithm

Criteria : "Time Quantum"

Mode : "Preemptive"

Process No	Arrival Time	Burst Time	Completion Time	TAT	WT	RT
	Time in ms	Time in ms	Time in ms	Time in ms	Time in ms	Time in ms
P <sub>1</sub>	0	5	12	12	12	0
P <sub>2</sub>	1	4	11	11	10	1
P <sub>3</sub>	2	2	6	6	4	2
P <sub>4</sub>	4	1	9	9	5	4

Ready Queue

Time Quantum	P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>	P <sub>1</sub>	P <sub>4</sub>	P <sub>2</sub>	P <sub>1</sub>
TQ = 2	← SFT						

Running Queue (Gantt Chart)

	P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>	P <sub>1</sub>	P <sub>4</sub>	P <sub>2</sub>	P <sub>1</sub>	
0	2	4	6	8	9	11	12	

Context Switching

( Sending Running Process back to Ready Queue  
and loading new program )

How

Context Switching

at (2, 4, 6, 8, 9, 11)

No. of Context Switching = 6

# • Preemptive Priority Scheduling Algorithm:

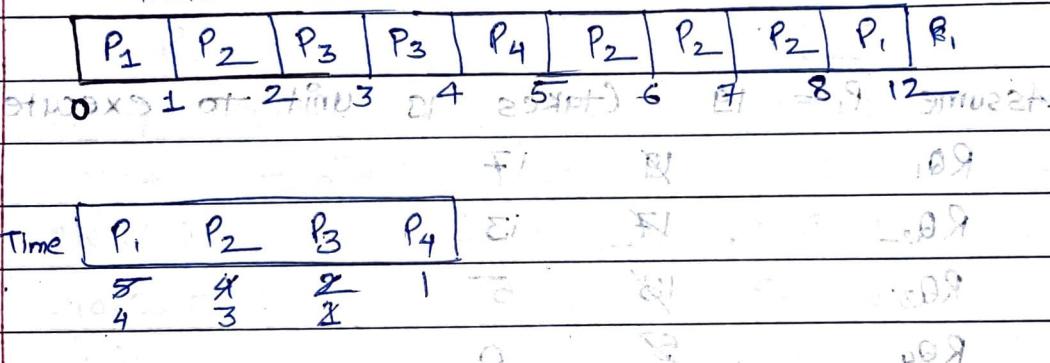
Criteria: "Priority"

Mode: "Preemptive"

Priority	Process	Arrival	Burst	Completion	TAT	WT	RT
		No	Time	Time	Time		
10	P <sub>1</sub>	0	5	12	12	7	
20	P <sub>2</sub>	4	4	8.9	8.9	3	
30	P <sub>3</sub>	2	2	8.9	8.9	0	
40	P <sub>4</sub>	4	1	5	1	0	

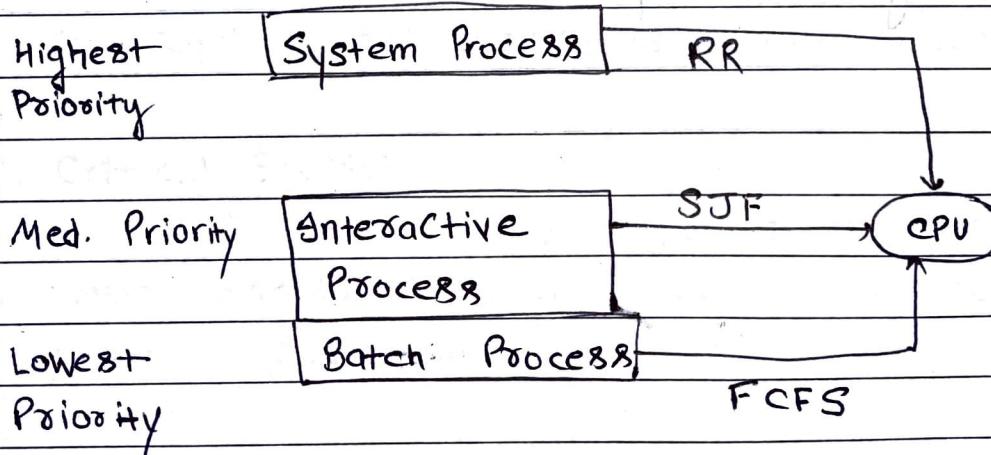
Higher the Value (Priority) → higher the Priority

Grantt Chart



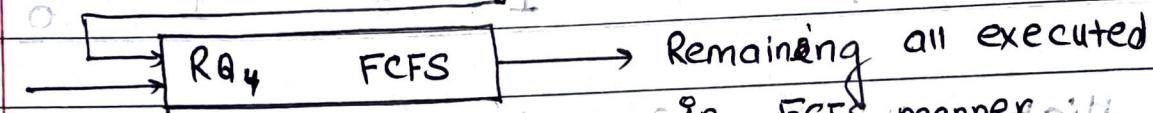
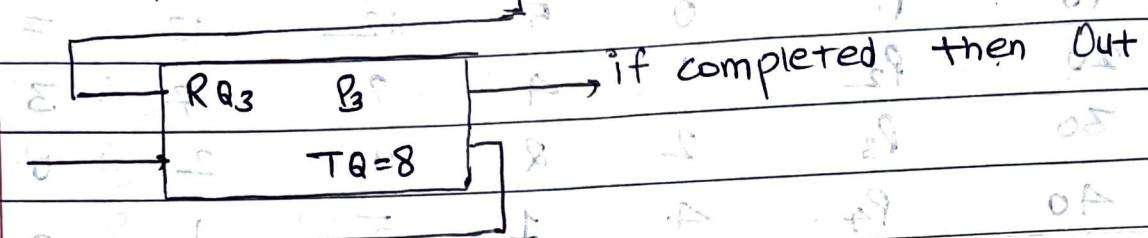
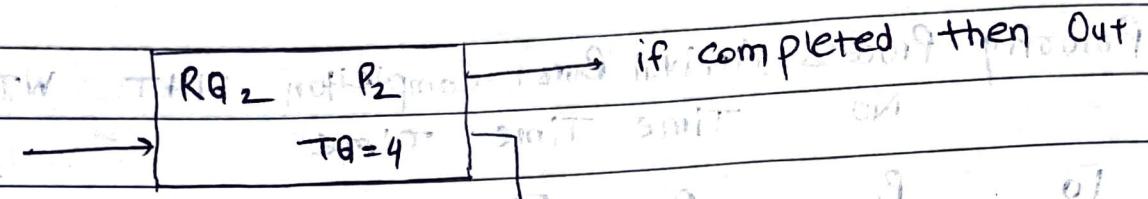
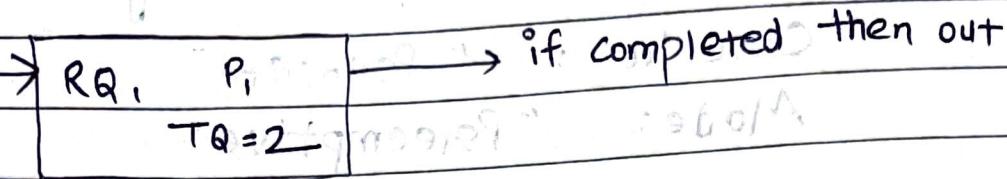
Impact between P & P (Priority Change)

Multi Level Queue Scheduling



Due to priority — Starvation

# Multilevel Feedback Queue Scheduling



Assume  $P_1 = 19$  (takes 19 unit to execute)

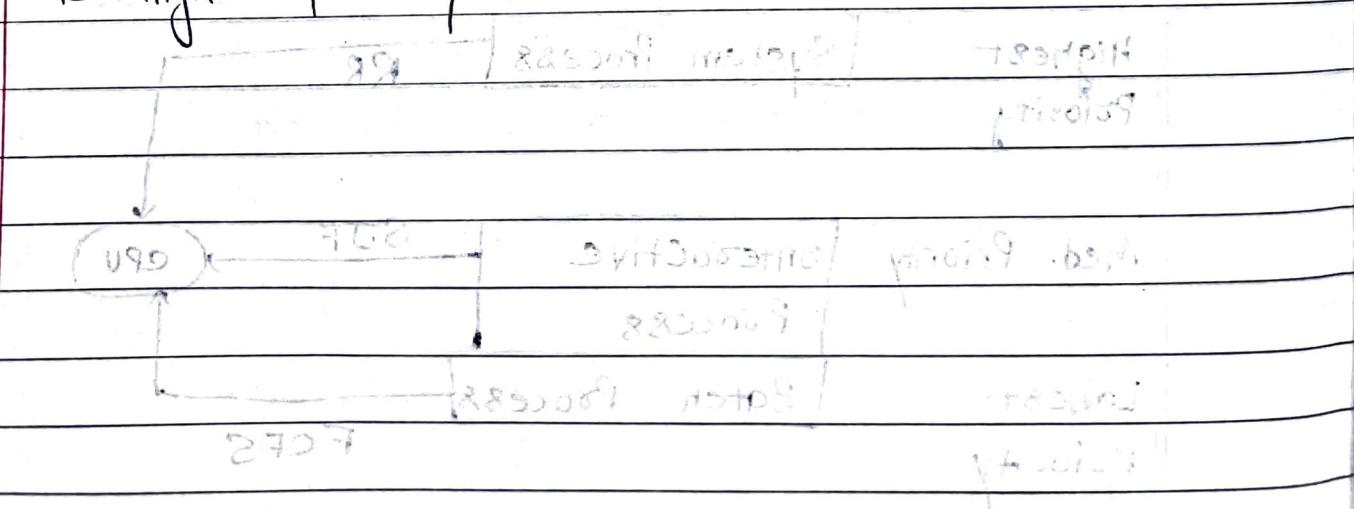
$RQ_1$       19      17

$RQ_2$       17      13

$RQ_3$       13      5

$RQ_4$       5      0

So the process will get executed for Time-Quantum  $TQ$  and if not completed then passed on to higher priority Queue.



which is utilizing of CPU