L 2.3

First come First Serve (FCFS) CPU Scheduling Algo with Example.

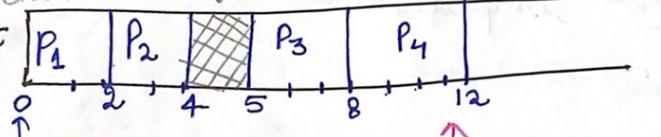
Process No.	Arrival Time	Burst Time (Execution Time)	Completion Time	TAT (C-A)	WT (TAT-BT)	RT (CPU First - AT)
P1	0	2	2	2	0	0-0 ⇒ 0
P2	1	2	4	3	1	2-1 ⇒ 1
P3	5	3	8	3	0	5-5 ⇒ 0
P4	6	4	12	6	2	8-6 ⇒ 2
		Useful Time for CPU	It is the time when P1, P2, P3, P4 got executed completely.	Actually for CPU	Actual for CPU — useful Time	

FCFS, Criteria = "Arival Time"

Mode = "Non-Preemptive".

CPU sits idle here bcoz here CPU has no process to execute.

Gantt chart



Start with Zero (P1 arrived at zero)

Time

at 12 all the processes got executed.

Avg TAT = $\frac{14}{4}$ = 3.5

Avg WT = $\frac{3}{4}$ = 0.75

0-0=0
2-1=1
5-5=0
8-6=2

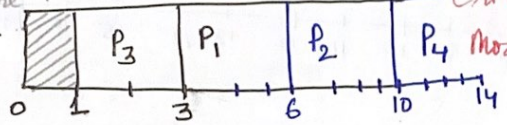
L-2.4.

Shortest Job First (SJF) Scheduling Algorithm with Example:-

Process No.	Arrival Time	Burst Time	Completion Time	TAT C-A	WT TAT-BT	RT CPU First - AT
P_1	1	3	6	5	2	3-1 ⇒ 2
P_2	2	4	10	8	4	6-2 ⇒ 4
P_3	1	2	3	2	0	1-1 ⇒ 0
P_4	4	4	14	10	6	10-4 ⇒ 6

CPU is Ideal here 1

Gantt chart



Critera = Burst time
Mode :- Non-preemptive

On Non-Pre-emptive RT = WT.

First BT then AT then Process Id / No

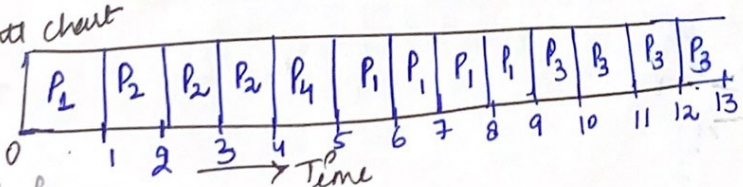
L-2.5.

Shortest Remaining Time First (SJF with Preemption Scheduling Algo:-

Critera :- Burst time, Mode :- Pre-emptive

Process No.	Arrival Time	Burst Time (Job Duration)	Completion Time	TAT C-A	WT TAT-BT	RT CPU first - A
P_1	0	~~5~~ 4	9	9-0=9	9-5=4	0-0=0
P_2	1	~~3~~ 1	4	4-1=3	3-3=0	1-1=0
P_3	2	4	13	13-2=11	11-4=7	9-2=7
P_4	4	1	5	5-4=1	1-1=0	4-4=0

Gantt chart



P_1 P_2 at 1 arrived

3 will execute this as contain less Burst time

L-27

Round Robin (RR) CPU Scheduling Algo. with Example:-

Process No.	Arrival Time	Burst Time	Completion Time	TAT (Compt - AT)	WT (TAT - BT)	RT (CPU first - AT)
P_1	0	~~5~~ 3 1	12	12	12-5 = 7	0-0 = 0
P_2	1	4 2	11	10	10-4 = 6	2-1 = 1
P_3	2	2 0	6	4	4-2 = 2	4-2 = 2
P_4	4	1 0	9	5	5-1 = 4	8-4 = 4

Criteria :- "Time - Quantum" , Mode :- "Pre-emptive"

Given Time Quantum = 2.

Context Switching :- Same seq. when one Process after Processed at particular Quantum getting back to Ready queue

→ Sequence of Processes in Ready Queue

Gantt chart

First check if any other process is ready to enter the Ready Queue then after that add the process from Running to Ready due to Context Switching

Depending upon the arrival time (sequence matters here)

RAM

Ready Queue: P_1 P_2 P_3 P_1 P_4 P_2 P_1

take the Ready process & put it in Running state

CPU

Running Queue: P_1 P_1 P_2 P_3 P_1 P_4 P_2 P_1

0 1 2 C.S 3 4 C.S 5 6 C.S 7 8 C.S 9 10 11 12

Not Cont. Swit

In all the Scheduling Algo.

Not context swi

Context Switching is happening here, (save the running process into PCB and Pick the new Process)

(No. of Context Switching = 6) we never consider the starting and ending bcoz there no saving & switching intentions)

Process Control Block (contains all the info related to that particular Process) • Advantage to resume the Process from the last one not to restart it from starting

PCB → save save father restart

L-2.8

Pre-emptive Priority Scheduling Algo. with Example:-

Priority	Process No	Arrival Time	Burst Time	Completion time	TAT (CT−AT)	WT (TAT−BT)
10	P_1	0	~~5~~ 4	12	12	7
20	P_2	1	~~4 3 2 1~~ 0	8	7	3
30	P_3	2	~~2 1~~ 0	4	2	0
40	P_4	4	~~1~~ 0	5	1	0

Criteria:- "Priority" Mode :- "Preemptive"

In Question It is given: **Higher the no. higher the priority**

Gantt chart

P_1	P_2	P_3	P_3	P_4	P_2	P_2	P_2	P_1	P_1	P_1	P_1
0–1	1–2	2–3	3–4	4–5	5–6	6–7	7–8	8–9	9–10	10–11	11–12

Step 1 Whatever the priority or criteria it is, run the process first who is arriving at 0.

Step 2:- at 1, P_2 arrives in Ready Queue and P_1 is already in running so now we will check the criteria of priority here

	P_1	P_2
Priority	10	20 ✓

Step 3:- at 2, Now P_3 arrives, here we will check the criteria

P_1	P_2	P_3
10	20	30

Step 4:- at 3, we have no new process in ready

P_1	P_2	P_3
10	20	30

Step 5:- at 4, check if we have new p. is Ready? yes then check —

P_1	P_2
10	

Remaining steps will be continue...

Imp NOTE:- If two processes have same priority then process will be picked depending on the Arrival time and If the arrival time is also same, go for the process number.

Example of mix Burst Time (CPU & I/O both) in CPU scheduling :-

Process	Arrival Time	Priority	First CPU	Second I/O	Third CPU	← Running Procedure completion Time
P_1	0	2	0 1 (CPU time means utne time tak chalana hai)	5	3 2 1	10
P_2	2	3	3 2 1 0	3	1	15
P_3	3	1	2 1 0	3	1 0	9
P_4	3	4	2 0	4	1	18

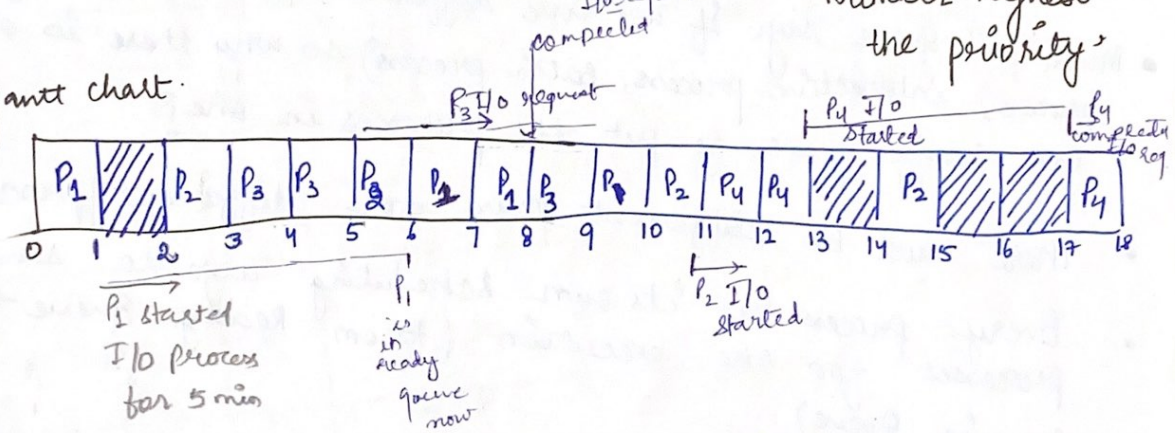
Burst Time

Mode :- Preemptive Criteria :- Priority based

↳ here : Lowest the Number highest the priority.

Find CT of P_1, P_2, P_3, P_4

Gantt chart.

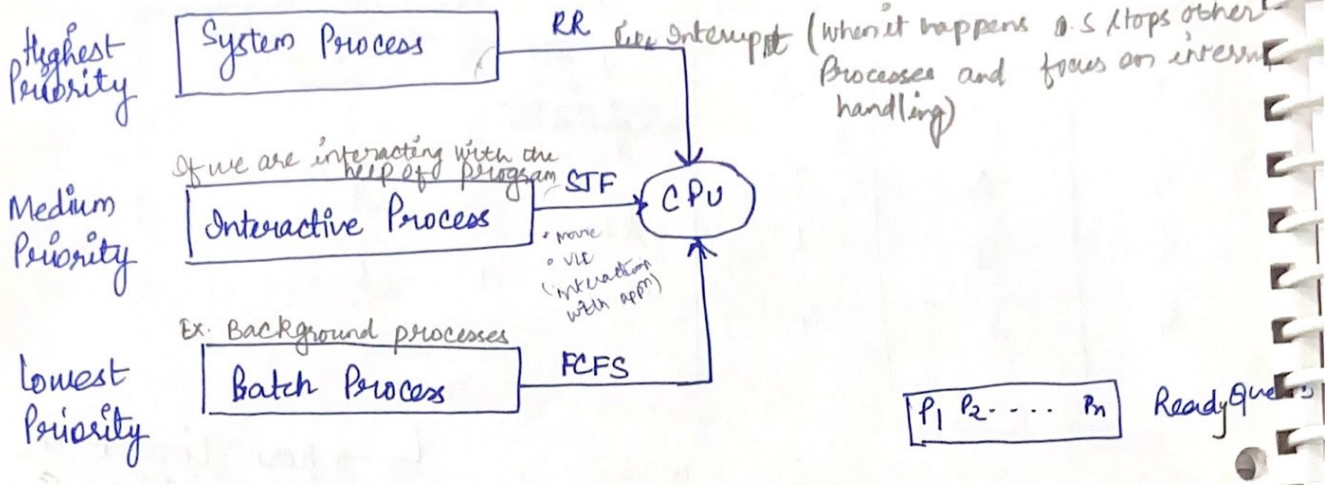


what is the ratio of CPU Idealness? $\frac{4}{18}$

" " " " " " usage? $\frac{14}{18}$

L-2.10

Multilevel Queue Scheduling :-



- Multilevel Queue says if we have different processes (ex. System process, interactive process, Batch process) so why there is only one ready queue to put the processes in one?
- There must be different Queue for different process.
- Every process has its own scheduling algo to send the processes for the execution (from Ready Queue to Running Queue)
- We are here categorizing the processes depending upon their real time scenario.

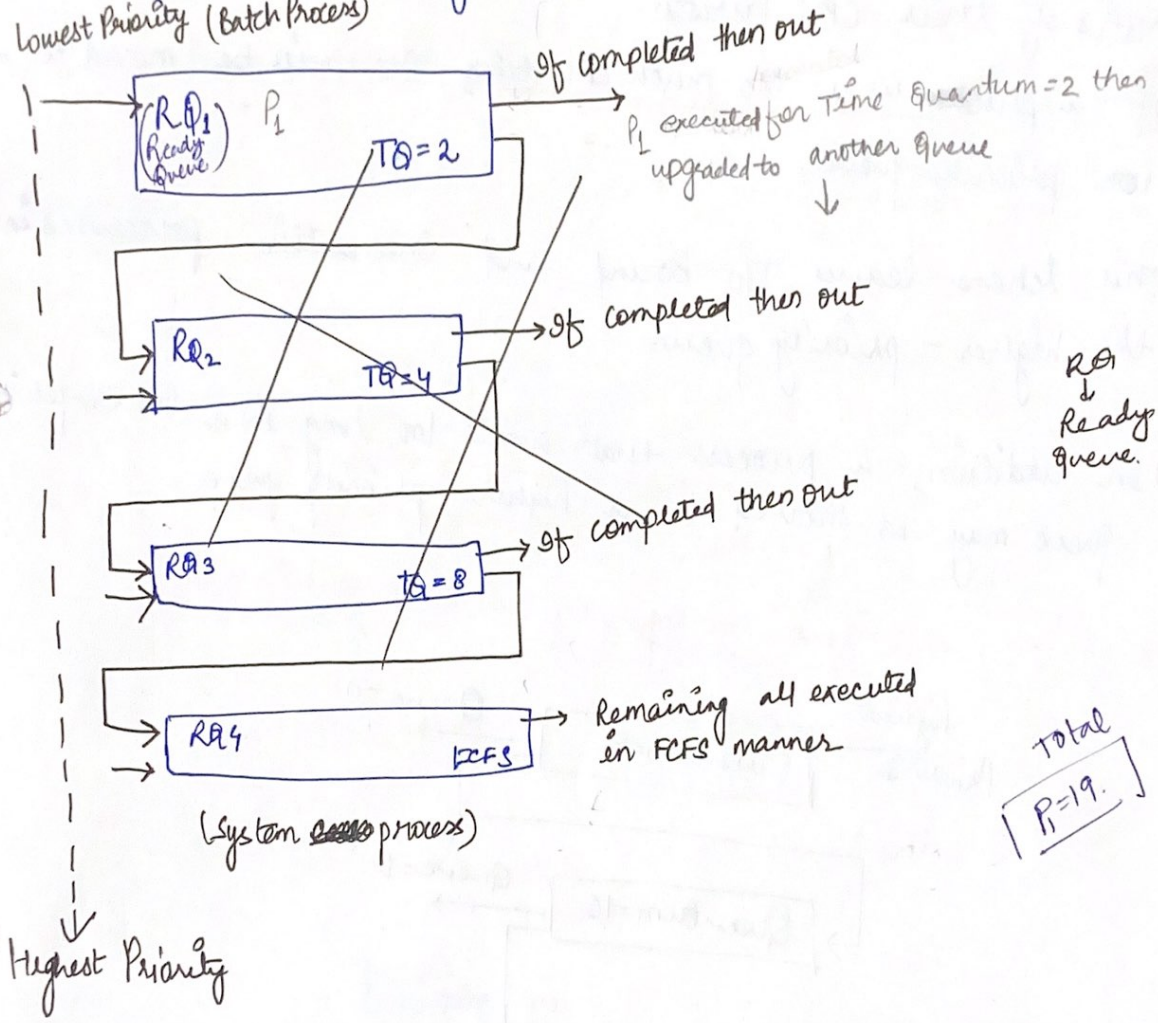
Prob:- If we have no. of system processes coming continuously then based upon their priority they will run before Interactive and Batch process which leads to wait for long for Interactive & batch processes

Prob → Starvation (Low priority processes will wait for s long for their turn)

Solu: Multilevel Feedback Queue

Multilevel Feedback Queue Scheduling :-

Lowest priority process gives the feedback, ~~back~~ based upon which ~~it~~ that process is being upgraded.



RQ → Ready Queue.

Total $P_1 = 19$.

- There is no need of Feedback for high priority process. because these tasks will come first everytime.

Note: This Scheduling is used for lower priority process (so that they have a chance of being executed) and to avoid starvation

the multilevel Feedback-queue scheduling Algo allow a process to move between queues.

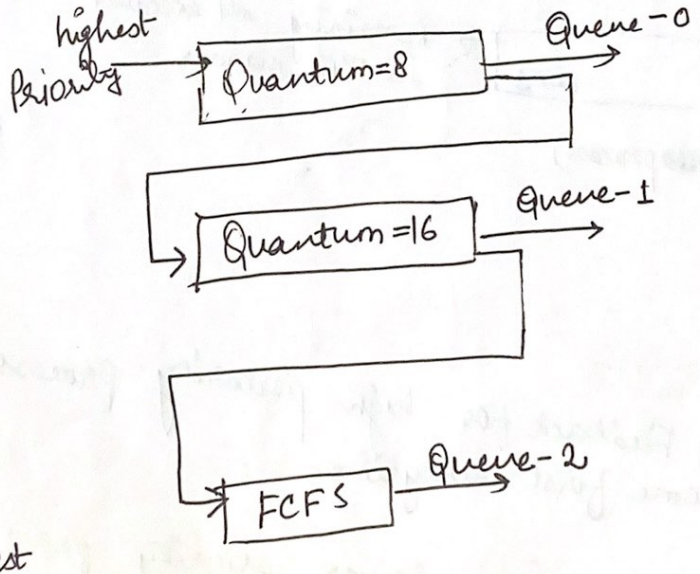
- the idea is to separate processes according to the characteristics of their CPU bursts.

- If a process uses too much CPU time, it will be moved to a low priority queue.

- This scheme leaves I/O bound and interactive processes in the higher-priority queue

- In addition, a process that waits too long in a lower-priority queue may be moved to a higher-priority queue.

∴ So this form of aging prevents starvation ∴



- Parameters depend
  - No. of queues
  - the scheduling algo for each queue
  - Method used to determine when to upgra a process to a higher priority queue
  - Method used to determine when to demo a process to a lower priority queue.
  - Method used to determine which queue a pr will enter when that process needs service.