Problems

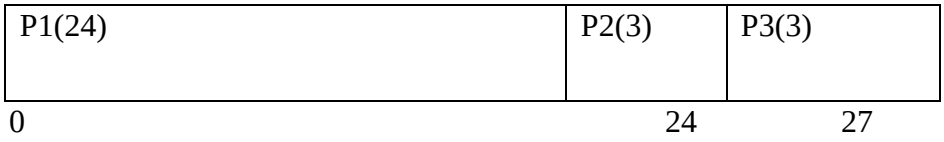
I) FCFS (First Come First Serve) Scheduling Algorithm

1) Find the Average Turnaround Time and Average Waiting Time

Process	Burst Time
P1	24
P2	3
P3	3

Gantt Chart:

Sequence order is P1--> P2-->P3



CT: Completion Time AT: Arrival Time BT : Burst Time

TAT : Turnaround –Time WT : Waiting-Time

TAT = CT – AT

WT = TAT - BT

Process	Burst Time	TAT	WT
P1	24	24-0=24	0
P2	3	27-0=27	24
P3	3	30-0=30	27
Total		**81**	**51**

Turnaround-Time

P1 TAT = CT – AT = 24 – 0 = 24

P2 TAT = CT – AT = 27 – 0 = 27

P3 TAT = CT – AT = 30 – 0 = 27

Waiting-Time

P1 WT = TAT – BT = 24 – 24= 0

P2 WT = TAT – BT = 27 – 03 = 24

P3 WT = TAT – BT = 30 – 03 = 25

Average TAT = Total TAT/No.of Processes = 81/3 = 27 ms

Average WT = Total WT/No.of Processes = 51/3 = 17 ms

Throughput = Total CT/ No. of. Processes = 30/3 = 10 ms

2) Find the Average Turnaround Time and Average Waiting Time

Process	Burst Time
P1	5
P2	24
P3	16
P4	10
P5	3

Gantt chart:

3) Find the Average Turnaround Time and Average Waiting Time

Process	Burst Time	Arrival Time
P1	7	0
P2	4	2
P3	1	4
P4	4	5

II) Shortest Job First (SJF) (SRTF – Shortest Remaining Time First)

Shortest Job First scheduling works on the process with the shortest **burst time** or **duration** first.

- This is the best approach to minimize waiting time.
- This is used in Batch Systems.
- It is of two types:
    1. Non Pre-emptive
    2. Pre-emptive
- To successfully implement it, the burst time/duration time of the processes should be known to the processor in advance, which is practically not feasible all the time.

- This scheduling algorithm is optimal if all the jobs/processes are available at the same time. (either Arrival time is 0 for all, or Arrival time is same for all)

i) Non Pre-emptive Shortest Job First

1) Find the Average Turnaround Time and Average Waiting Time

Process	Burst Time
P1	24
P2	3
P3	3

2) Find the Average Turnaround Time and Average Waiting Time

Process	Burst Time
P1	6
P2	8
P3	7
P4	3

3) Find the Average Turnaround Time and Average Waiting Time

Process	Burst Time
P1	21
P2	3
P3	6
P4	2

i) Pre-emptive Shortest Job First

1) Find the Average Turnaround Time and Average Waiting TimeUsing Preemptive

Process	Burst Time	Arrival Time
P1	7	0
P2	4	2
P3	1	4
P4	4	5

2) Find the Average Turnaround Time and Average Waiting Time

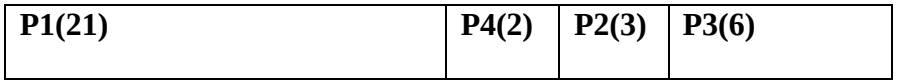
I) Using Non-Preemptive (sjf)

ii) Using Preemptive (SRTF)

Process	Burst Time	Arrival Time
P1	21	0
P2	3	1
P3	6	2
P4	2	3

I) Using Non-Preemptive (sjf)

GANTT CHART:



0 **21** **23** **26** **32**

Process	Burst Time	Arrival Time	TAT(CT-AT)	WT(TAT-BT)
P1	21	0	21-0=21	21-21=0
P2	3	1	26-1=25	25-3=22
P3	6	2	32-2=30	30-6=24
P4	2	3	23-3=20	20-2=18
TOTAL			96	64

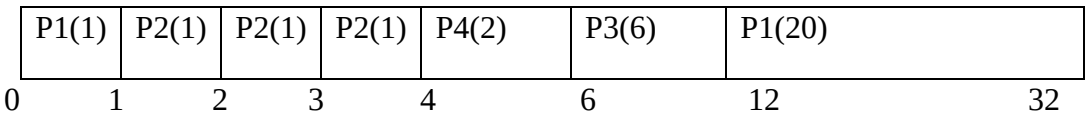
Average TAT = Total TAT/ No.of Processes = 96 / 4 = 24 ms

Average WT = Total WT/ No.of Processes = 64 / 4 = 16 ms

Throughput = 32 / 4 = 8 ms

ii) Using Preemptive (SRTF)

Gantt Chart:



0 1 2 3 4 6 12 32

After 1 ms P2 has entered into Ready Queue.

P1(20), P2(3)

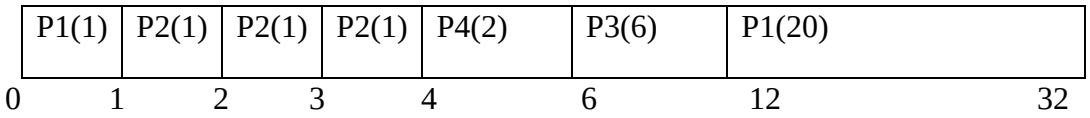
After 2 ms P3 has entered into Ready Queue.

P1(20), P2(2),P3(6)

After 3 ms P4 has entered into Ready Queue

P1(20), P2(1),P3(6), P4(2)

P1(20), P3(6), P4(2)



Process	Burst Time	Arrival Time	TAT(CT-AT)	WT(TAT-BT)
P1	21	0	32-0=32	32-21=11
P2	3	1	4-1=3	3-3=0
P3	6	2	12-2=10	10-6=4
P4	2	3	6-3=3	3-2=1
TOTAL			48	16

Average TAT = Total TAT/ No.of Processes = 48 / 4 = 12 ms

Average WT = Total WT/ No.of Processes = 16 / 4 = 4 ms

Throughput = 32 / 4 = 8 ms

Scheduling Al;gorithm	**Average TAT**	**Average WT**
FCFS		
Non-Premptive SJF	**24**	**16**

SRTF(Preemptive)	**12**	**4**

3) Find the Average Turnaround Time and Average Waiting Time

I) Using Non-Preemptive

ii) Using Preemptive (SRTF)

Process	Burst Time	Arrival Time
P1	8	0
P2	4	1
P3	9	2
P4	5	3

I) Using Non-Preemptive

P1(8)	**P2(4)**	**P4(5)**	**P3(9)**

0 **8** **12** **17** **26**

Process	Burst Time	Arrival Time	TAT(CT-AT)	WT(TAT-BT)
P1	8	0	8-0 = 8	8-8=0
P2	4	1	12-1=11	11-4=7
P3	9	2	26-2=24	24-9=15
P4	5	3	17-3=14	14-5=9
TOTAL			57	31

Average TAT = Total TAT/ No.of Processes = 57/4 = 14.25 ms

Average WT = Total WT/ No.of Processes = 31 / 4 = 7.75 ms

Throughput = 32 / 4 = 8 ms

I) Using Preemptive

P1(1)	**P2(1)**	**P2(1)**	**P2(2)**	**P4(5)**	**P1(7)**	**P3(9)**

0 **1** **2** **3** **5** **10** **17**

26

Process	Burst Time	Arrival Time	TAT(CT-AT)	WT(TAT-BT)
P1	8	0		
P2	4	1		
P3	9	2		
P4	5	3		
TOTAL				

Disadvantages:

Starvation:

Priority Scheduling Algorithm

In the Shortest Job First scheduling algorithm, the priority of a process is generally the inverse of the CPU burst time, i.e. the larger the burst time the lower is the priority of that process.

In case of priority scheduling the priority is not always set as the inverse of the CPU burst time, rather it can be internally or externally set, but yes the scheduling is done on the basis of priority of the process where the process which is most urgent is processed first, followed by the ones with lesser priority in order.

Processes with same priority are executed in FCFS manner.

The priority of process, when internally defined, can be decided based on **memory requirements**, **time limits**, **number of open files**, **ratio of I/O burst to CPU burst** etc.

Whereas, external priorities are set based on criteria outside the operating system, like the importance of the process and funds paid for the computer resource use.

Priority Numbers Range is : 0 to 7 or 0 to 4096

0 –Highest Priority

7 – Lowest Priority

Types of Priority Scheduling Algorithm

Priority scheduling can be of two types:

1. **Non-Preemptive Priority Scheduling**: In case of non-preemptive priority scheduling algorithm if a new process arrives with a higher priority than the current running process, the incoming process is put at the head of the ready queue, which means after the execution of the current process it will be processed.
2. **Preemptive Priority Scheduling**: If the new process arrived at the ready queue has a higher priority than the currently running process, the CPU is preempted, which means

the processing of the current process is stopped and the incoming new process with higher priority gets the CPU for its execution.

Problems

1)

Process	Burst Time	Priority
P1	21	2
P2	3	1
P3	6	4
P4	2	3

2)

Process	Burst Time	Priority	Arrival Time
P1	10	3	0
P2	1	1	2
P3	2	3	3
P4	1	4	1
P5	5	2	1

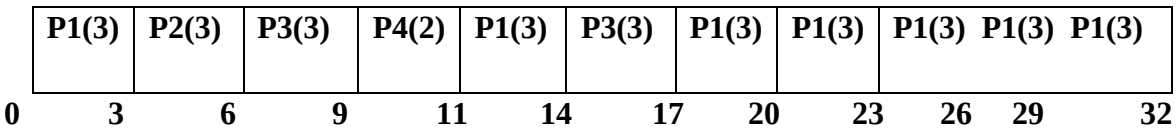
Round-Robin Scheduling Algorithms

- It is similar to FCFS Scheduling Algorithm.
- It is preemptive.
- A fixed time is allotted to each process, called quantum or time-slice , for execution.
- Once a process is executed for given time period that process is preempted and other process executes for given time period.
- Context switching is used to save states of preemptied processes.

Process	Burst Time
P1	21
P2	3
P3	6
P4	2

Time –slice = 3 ms

Gantt Chart:



Process	Burst Time	TAT(CT-AT)	WT(TAT-BT)
P1	21	32-0= 32	32-21= 11
P2	3	6-0= 6	6-3=3
P3	6	17=0=17	17-6=11
P4	2	11-0=11	11-2=9
TOTAL		66	34

Average TAT = 66/4 = 16.5 ms

Average WT = 34 / 4 = 8. 5 ms

MULTI-LELEL QUEUE SCHEDULING ALGORITHM

A multi-level queue scheduling algorithm partitions the ready queue into several separate queues. The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority, or process type. Each queue has its own scheduling algorithm.

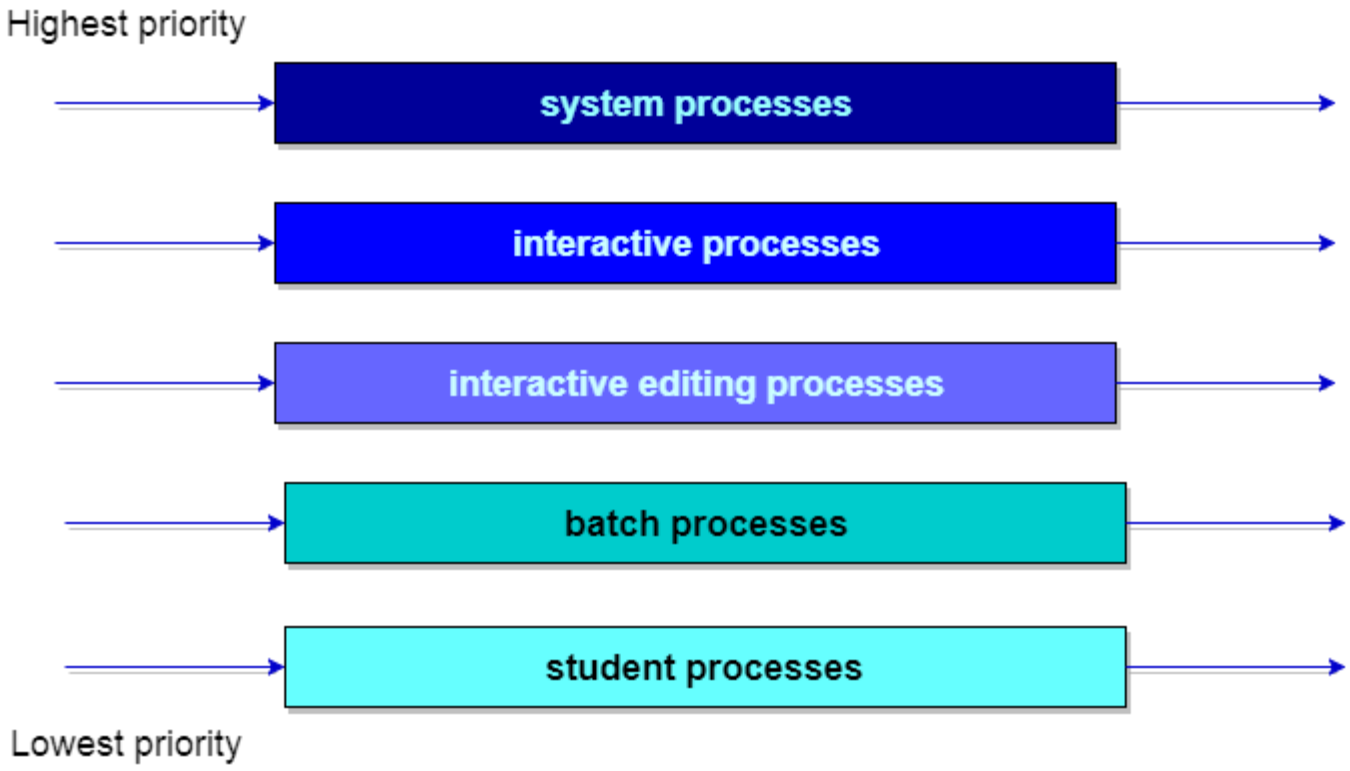
For example: separate queues might be used for foreground and background processes. The foreground queue might be scheduled by Round Robin algorithm, while the background queue is scheduled by an FCFS algorithm.

In addition, there must be scheduling among the queues, which is commonly implemented as fixed-priority preemptive scheduling. **For example:** The foreground queue may have absolute priority over the background queue.

Let us consider an example of a multilevel queue-scheduling algorithm with five queues:

1. System Processes
2. Interactive Processes
3. Interactive Editing Processes
4. Batch Processes
5. Student Processes

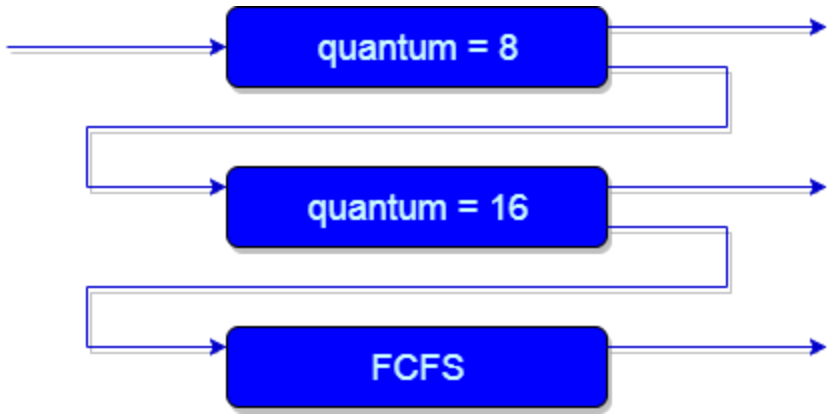
Each queue has absolute priority over lower-priority queues. No process in the batch queue, for example, could run unless the queues for system processes, interactive processes, and interactive editing processes were all empty. If an interactive editing process entered the ready queue while a batch process was running, the batch process will be preempted.



Multilevel Feedback Queue Scheduling

In a multilevel queue-scheduling algorithm, processes are permanently assigned to a queue on entry to the system. Processes do not move between queues. This setup has the advantage of low scheduling overhead, but the disadvantage of being inflexible.

Multilevel feedback queue scheduling, however, allows a process to move between queues. The idea is to separate processes with different CPU-burst characteristics. If a process uses too much CPU time, it will be moved to a lower-priority queue. Similarly, a process that waits too long in a lower-priority queue may be moved to a higher-priority queue. This form of aging prevents starvation.



An example of a multilevel feedback queue can be seen in the below figure.

In general, a multilevel feedback queue scheduler is defined by the following parameters:

- The number of queues.
- The scheduling algorithm for each queue.
- The method used to determine when to upgrade a process to a higher-priority queue.
- The method used to determine when to demote a process to a lower-priority queue.
- The method used to determine which queue a process will enter when that process needs service.

The definition of a multilevel feedback queue scheduler makes it the most general CPU-scheduling algorithm. It can be configured to match a specific system under design. Unfortunately, it also requires some means of selecting values for all the parameters to define the best scheduler. Although a multilevel feedback queue is the **most general scheme**, it is also the **most complex**.