Lecture – 1   Operating System Concept (Take lecture from power point)
Lecture – 2   Different Scheduling Algorithms ::

First Come First Serve (FCFS): Simplest scheduling algorithm that schedules according to arrival times of processes. First come first serve scheduling algorithm states that the process that requests the CPU first is allocated the CPU first. It is implemented by using the FIFO queue. When a process enters the ready queue, its PCB is linked onto the tail of the queue. When the CPU is free, it is allocated to the process at the head of the queue. The running process is then removed from the queue. FCFS is a non-preemptive scheduling algorithm.
Note:First come first serve suffers from convoy effect.
Shortest Job First (SJF): Process which have the shortest burst time are scheduled first.If two processes have the same bust time then FCFS is used to break the tie. It is a non-preemptive scheduling algorithm.
executing, can't be interrupted before complete execution.
Shortest Remaining Time First (SRTF): It is preemptive mode of SJF algorithm in which jobs are schedule according to shortest remaining time.
Longest Remaining Time First (LRTF): It is preemptive mode of LJF algorithm in which we give priority to the process having largest burst time remaining.
Round Robin Scheduling: Each process is assigned a fixed time(Time Quantum/Time Slice) in cyclic way.It is designed especially for the time-sharing system. The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1-time quantum. To implement Round Robin scheduling, we keep the ready queue as a FIFO queue of processes. New processes are added to the tail of the ready queue. The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1-time quantum, and dispatches the process. One of two things will then happen. The process may have a CPU burst of less than 1-time quantum. In this case, the process itself will release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue. Otherwise, if the CPU burst of the currently running process is longer than 1-time quantum, the timer will go off and will cause an interrupt to the operating system. A context switch will be executed, and the process will be put at the tail of the ready queue. The CPU scheduler will then select the next process in the ready queue.
Priority Based scheduling (Non-Preemptive): In this scheduling, processes are scheduled according to their priorities, i.e., highest priority process is scheduled first. If priorities of two processes match, then schedule according to arrival time.   Here starvation of process is possible.
Highest Response Ratio Next (HRRN): In this scheduling, processes with highest response ratio is scheduled. This algorithm avoids starvation.
Response Ratio = (Waiting Time + Burst time) / Burst time
Multilevel Queue Scheduling: According to the priority of process, processes are placed in the different queues. Generally high priority process are placed in the top level queue. Only after completion of processes from top level queue, lower level queued processes are scheduled. It can suffer from starvation.
Multi level Feedback Queue Scheduling: It allows the process to move in 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 is moved to a lower-priority queue.

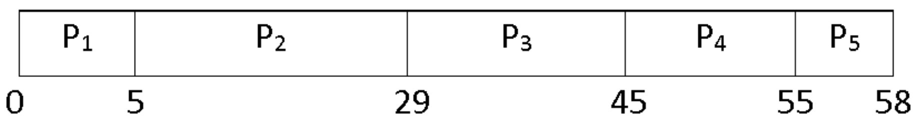
Some useful facts about Scheduling Algorithms: ::

1. FCFS can cause long waiting times, especially when the first job takes too much CPU time.
2. Both SJF and Shortest Remaining time first algorithms may cause starvation. Consider a situation when the long process is there in the ready queue and shorter processes keep coming.
3. If time quantum for Round Robin scheduling is very large, then it behaves same as FCFS scheduling.
4. SJF is optimal in terms of average waiting time for a given set of processes ,i.e., average waiting time is minimum with this scheduling, but problems are, how to know/pre CPU scheduling algorithm and its problems :

Problems based on CPU scheduling ::

Process	Burst Time(ms)
P_1	5
P_2	24
P_3	16
P_4	10
P_5	3

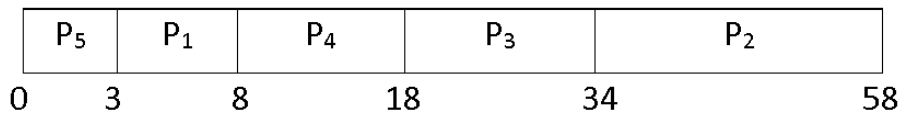
::



FCFS SCHEDULING

Average WT=25 ms   Average TAT = 192/5=38.4 ms Throughput=(5+24+16+10+3)/5=11.6ms

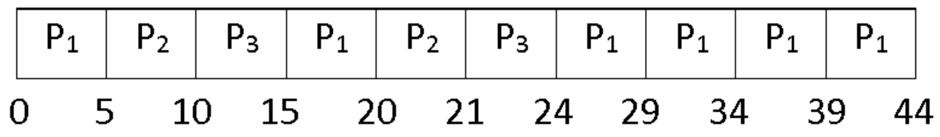
Process	Burst Time(ms)
P_1	5
P_2	24
P_3	16
P_4	10
P_5	3



SJF scheduling: AWT=12.6ms   ATAT=24.2ms   Throughput=11.6 ms

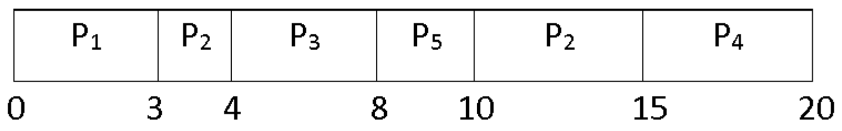
Process	CPU Burst Time
P_1	30
P_2	6
P_3	8

Gantt Chart



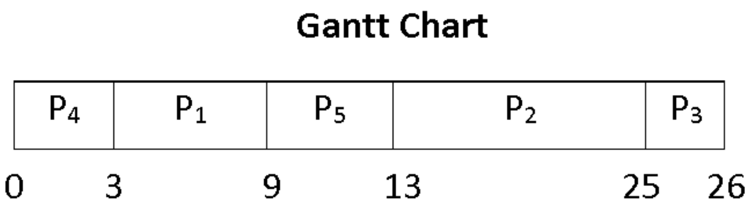
ROUND ROBIN scheduling : AWT=15ms   ATAT=29.66ms   Throughput=14.66 ms

Process	Burst Time(CPU)	Arrival Time(ms)
P_1	3	0
P_2	6	2
P_3	4	4
P_4	5	6
P_5	2	8



SRTF scheduling : AWT=3.2ms   ATAT=7.2ms   Throughput=4 ms

Process	CPU Burst Time	Priority
P_1	6	2
P_2	12	4
P_3	1	5
P_4	3	1
P_5	4	3



Priority scheduling : AWT=10ms ATAT=15.2ms Throughput=5.2 ms

Problem 1: Consider the following table of arrival time and burst time for three processes P0, P1 and P2.

Process	Arrival time	Burst Time
P0	0 ms	9 ms
P1	1 ms	4 ms
P2	2 ms	9 ms

The pre-emptive shortest job first scheduling algorithm is used. Scheduling is carried out only at arrival or completion of processes. What is the average waiting time for the three processes?
(A) 5.0 ms (B) 4.33 ms (C) 6.33 (D) 7.33 Solution : Answer: – (A)

Process P0 is allocated processor at 0 ms as there is no other process in the ready queue. P0 is preempted after 1 ms as P1 arrives at 1 ms and burst time for P1 is less than remaining time of P0. P1 runs for 4ms. P2 arrived at 2 ms but P1 continued as burst time of P2 is longer than P1. After P1 completes, P0 is scheduled again as the remaining time for P0 is less than the burst time of P2. P0 waits for 4 ms, P1 waits for 0 ms and P2 waits for 11 ms. So average waiting time is

(0+4+11)/3 = 5.

Problem2 :Consider the following set of processes, with the arrival times and the CPU-burst times given in milliseconds

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

What is the average turnaround time for these processes with the preemptive shortest remaining processing time first (SRPT / SRTF) algorithm ?
(A) 5.50 (B) 5.75 (C) 6.00 (D) 6.25 Answer (A)

Solution:
The following is Gantt Chart of execution

P1	P2	P4	P3	P1
1	4	5	8	12

Turn Around Time = Completion Time – Arrival Time
Avg Turn Around Time = (12 + 3 + 6+ 1)/4 = 5.50

Problem 3: An operating system uses the Shortest Remaining Time first (SRTF) process scheduling algorithm. Consider the arrival times and execution times for the following processes:

Process	Execution time	Arrival time
P1	20	0
P2	25	15
P3	10	30
P4	15	45

What is the total waiting time for process P2?
(A) 5 (B) 15 (C) 40 (D) 55 Answer (B)
At time 0, P1 is the only process, P1 runs for 15 time units.
At time 15, P2 arrives, but P1 has the shortest remaining time. So P1 continues for 5 more time units.
At time 20, P2 is the only process. So it runs for 10 time units
At time 30, P3 is the shortest remaining time process. So it runs for 10 time units
At time 40, P2 runs as it is the only process. P2 runs for 5 time units.
At time 45, P3 arrives, but P2 has the shortest remaining time. So P2 continues for 10 more time units.
P2 completes its execution at time 55

Total waiting time for P2 = Completion time - (Arrival time + Execution time)

= 55 - (15 + 25)

= 15

Problem-04:

Consider three process, all arriving at time zero, with total execution time of 10, 20 and 30 units respectively. Each process spends the first 20% of execution time doing I/O, the next 70% of time doing computation, and the last 10% of time doing I/O again. The operating system uses a shortest remaining compute time first scheduling algorithm and schedules a new process either when the running process gets blocked on I/O or when the running process finishes its compute burst. Assume that all I/O operations can be overlapped as much as possible. For what percentage of does the CPU remain idle?

1. 0% 2. 10.6% 3. 30.0% 4. 89.4%

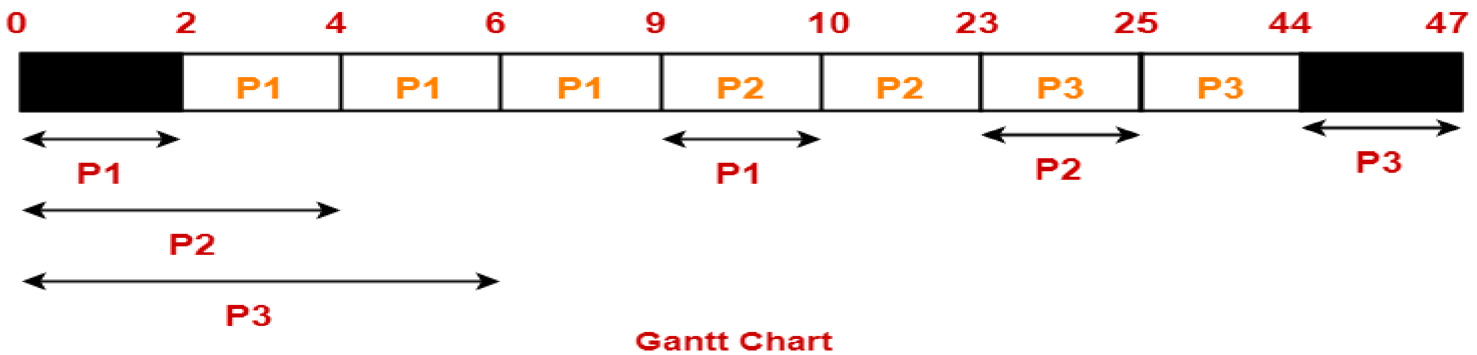
Solution-

According to question, we have-

	Total Burst Time	I/O Burst	CPU Burst	I/O Burst
Process P1	10	2	7	1
Process P2	20	4	14	2
Process P3	30	6	21	3

The scheduling algorithm used is Shortest Remaining Time First.

Gantt Chart-



Gantt Chart

Percentage of time CPU remains idle

= (5 / 47) x 100

= 10.638%

Thus, Option (B) is correct.

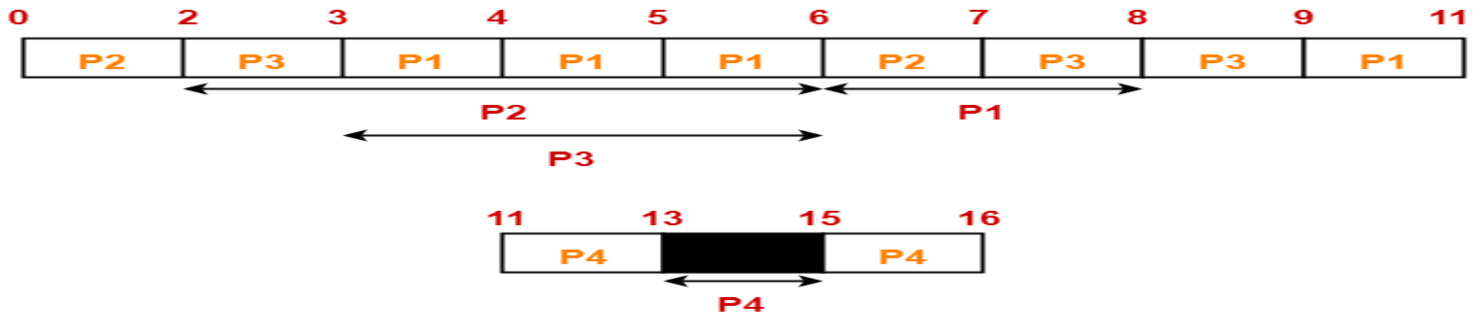
Problem-05:

Consider the set of 4 processes whose arrival time and burst time are given below-

Process No.	Arrival Time	Burst Time		
		CPU Burst	I/O Burst	CPU Burst
P1	0	3	2	2
P2	0	2	4	1
P3	2	1	3	2
P4	5	2	2	1

If the CPU scheduling policy is Shortest Remaining Time First, calculate the average waiting time and average turn around time.

Solution- Gantt Chart-



Gantt Chart

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Also read- Various Times Of Process

Process Id	Exit time	Turn Around time	Waiting time
P1	11	11 – 0 = 11	11 – (3+2) = 6
P2	7	7 – 0 = 7	7 – (2+1) = 4
P3	9	9 – 2 = 7	7 – (1+2) = 4
P4	16	16 – 5 = 11	11 – (2+1) = 8

- Average Turn Around time = (11 + 7 + 7 + 11) / 4 = 36 / 4 = 9 units
- Average waiting time = (6 + 4 + 4 + 8) / 4 = 22 / 5 = 4.4 units

Problem-03:

Consider the set of 3 processes whose arrival time and burst time are given below-

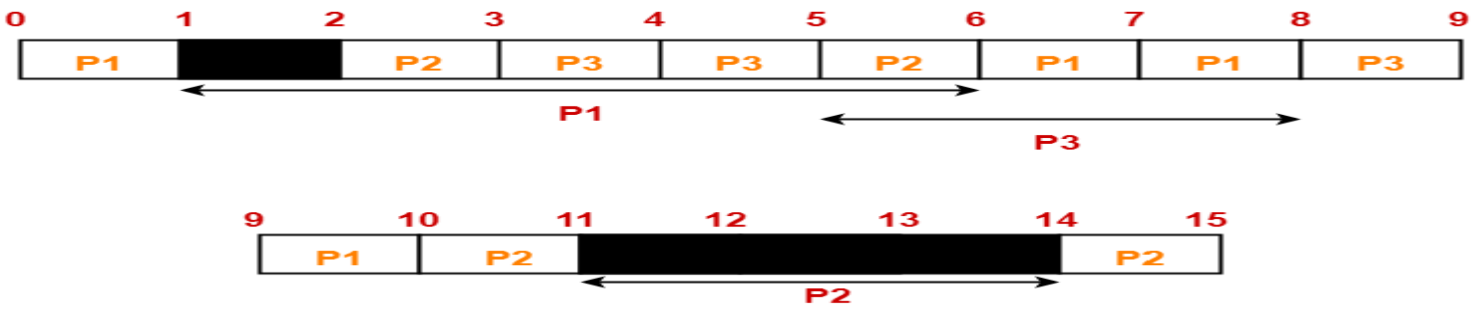
Process No.	Arrival Time	Priority	Burst Time		
			CPU Burst	I/O Burst	CPU Burst
P1	0	2	1	5	3
P2	2	3	3	3	1
P3	3	1	2	3	1

If the CPU scheduling policy is Priority Scheduling, calculate the average waiting time and average turn around time. (Lower number means higher priority)

Solution-

The scheduling algorithm used is Priority Scheduling.

Gantt Chart-



Gantt Chart

Turn Around time = Exit time – Arrival time

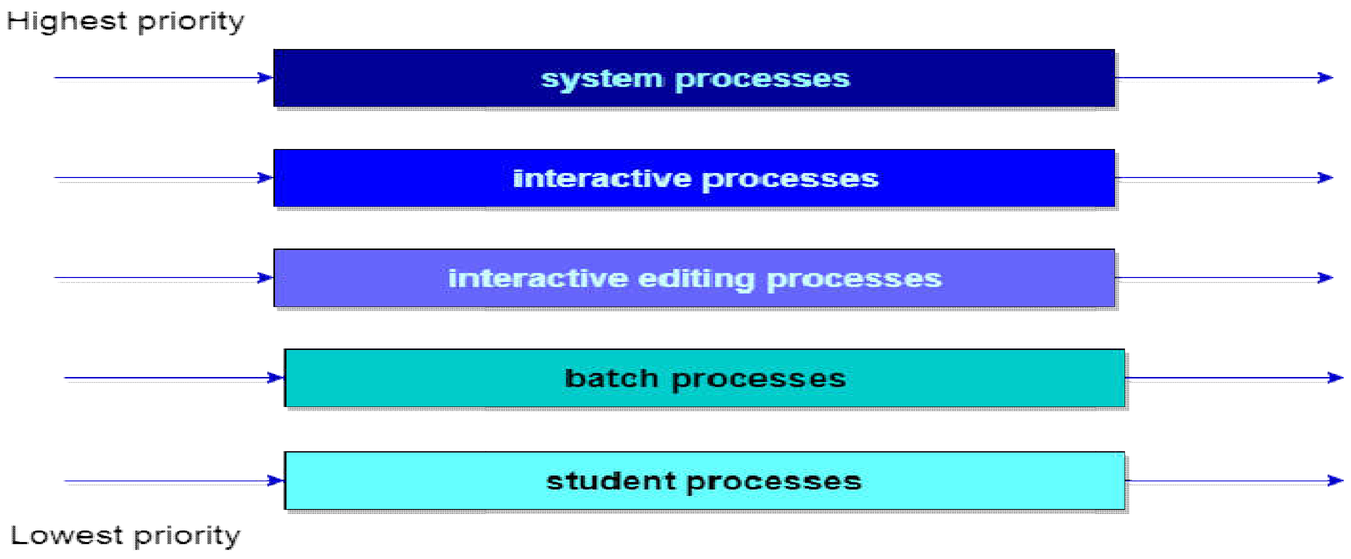
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	10	10 – 0 = 10	10 – (1+3) = 6
P2	15	15 – 2 = 13	13 – (3+1) = 9
P3	9	9 – 3 = 6	6 – (2+1) = 3

Average Turn Around time = (10 + 13 + 6) / 3 = 29 / 3 = 9.67 units

- Average waiting time = (6 + 9 + 3) / 3 = 18 / 3 = 6 units

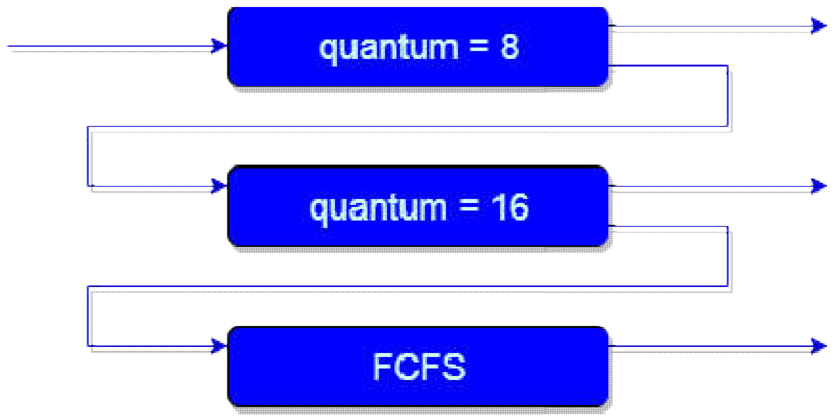
A **multi-level queue scheduling** algorithm partitions the ready **queue** into several separate **queues**. ... The processes are permanently assigned to one **queue**, generally based on some property of the process, such as memory size, process priority, or process type.



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.

Comparison of Scheduling Algorithms

By now, you must have understood how CPU can apply different scheduling algorithms to schedule processes. Now, let us examine the advantages and disadvantages of each scheduling algorithms that we have studied so far.

First Come First Serve (FCFS)

Let's start with the **Advantages:**

- FCFS algorithm doesn't include any complex logic, it just puts the process requests in a queue and executes it one by one.
- Hence, FCFS is pretty simple and easy to implement.
- Eventually, every process will get a chance to run, so starvation doesn't occur.

It's time for the **Disadvantages:**

- There is no option for pre-emption of a process. If a process is started, then CPU executes the process until it ends.
- Because there is no pre-emption, if a process executes for a long time, the processes in the back of the queue will have to wait for a long time before they get a chance to be executed.

Shortest Job First (SJF)

Starting with the **Advantages:** of Shortest Job First scheduling algorithm.

- According to the definition, short processes are executed first and then followed by longer processes.
- The throughput is increased because more processes can be executed in less amount of time.

And the **Disadvantages:**
The time taken by a process must be known by the CPU beforehand, which is not possible.

- Longer processes will have more waiting time, eventually they'll suffer starvation.

Note: Preemptive Shortest Job First scheduling will have the same advantages and disadvantages as those for SJF.

Round Robin (RR)

Here are some **Advantages:** of using the Round Robin Scheduling:

- Each process is served by the CPU for a fixed time quantum, so all processes are given the same priority.
- Starvation doesn't occur because for each round robin cycle, every process is given a fixed time to execute. No process is left behind.

and here comes the **Disadvantages:**

- The throughput in RR largely depends on the choice of the length of the time quantum. If time quantum is longer than needed, it tends to exhibit the same behavior as FCFS.
- If time quantum is shorter than needed, the number of times that CPU switches from one process to another process, increases. This leads to decrease in CPU efficiency.

Priority based Scheduling

Advantages of Priority Scheduling:

- The priority of a process can be selected based on memory requirement, time requirement or user preference. For example, a high end game will have better graphics, that means the process which updates the screen in a game will have higher priority so as to achieve better graphics performance.

Some **Disadvantages:**

- A second scheduling algorithm is required to schedule the processes which have same priority.

- In preemptive priority scheduling, a higher priority process can execute ahead of an already executing lower priority process. If lower priority process keeps waiting for higher priority processes, starvation occurs.

Usage of Scheduling Algorithms in Different Situations

Every scheduling algorithm has a type of a situation where it is the best choice. Let's look at different such situations:

Situation 1:

The incoming processes are short and there is no need for the processes to execute in a specific order.

In this case, FCFS works best when compared to SJF and RR because the processes are short which means that no process will wait for a longer time. When each process is executed one by one, every process will be executed eventually.

Situation 2:

The processes are a mix of long and short processes and the task will only be completed if all the processes are executed successfully in a given time.

Round Robin scheduling works efficiently here because it does not cause starvation and also gives equal time quantum for each process.

Situation 3:

The processes are a mix of user based and kernel based processes.

Priority based scheduling works efficiently in this case because generally kernel based processes have higher priority when compared to user based processes.

For example, the scheduler itself is a kernel based process, it should run first so that it can schedule other processes.

Shortest Remaining Time First (SRTF) Scheduling Algorithm

This Algorithm is the **preemptive version** of **SJF scheduling**. In SRTF, the execution of the process can be stopped after certain amount of time. At the arrival of every process, the short term scheduler schedules the process with the least remaining burst time among the list of available processes and the running process.

Once all the processes are available in the **ready queue**, No preemption will be done and the algorithm will work as **SJF scheduling**. The context of the process is saved in the **Process Control Block** when the process is removed from the execution and the next process is scheduled. This PCB is accessed on the **next execution** of this process.

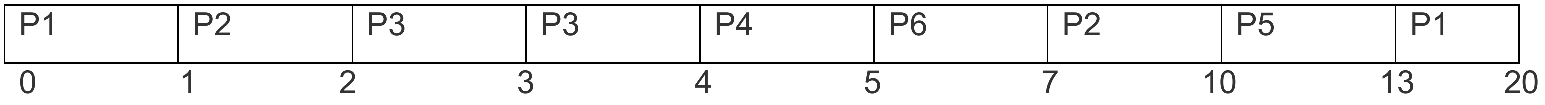
Example

In this Example, there are five jobs P1, P2, P3, P4, P5 and P6. Their arrival time and burst time are given below in the table. Use SRTF scheduling algorithm calculate the average waiting time for the above snapshot.

Process	Arrival	Burst	Completion	Turn Around	Waiting	Response

ID	Time	Time	Time	Time	Time	Time
1	0	8	20	20	12	0
2	1	4	10	9	5	1
3	2	2	4	2	0	2
4	3	1	5	2	1	4
5	4	3	13	9	6	10
6	5	2	7	2	0	5

Gantt Chart of SRTF scheduling::



Avg Waiting Time = 24/6 = 4ms

The Gantt chart is prepared according to the arrival and burst time given in the table.

1. Since, at time 0, the only available process is P1 with CPU burst time 8. This is the only available process in the list therefore it is scheduled.
2. The next process arrives at time unit 1. Since the algorithm we are using is SRTF which is a preemptive one, the current execution is stopped and the scheduler checks for the process with the least burst time. Till now, there are two processes available in the ready queue. The OS has executed P1 for one unit of time till now; the remaining burst time of P1 is 7 units. The burst time of Process P2 is 4 units. Hence Process P2 is scheduled on the CPU according to the algorithm.
3. The next process P3 arrives at time unit 2. At this time, the execution of process P3 is stopped and the process with the least remaining burst time is searched. Since the process P3 has 2 unit of burst time hence it will be given priority over others.
4. The Next Process P4 arrives at time unit 3. At this arrival, the scheduler will stop the execution of P4 and check which process is having least burst time among the available processes (P1, P2, P3 and P4). P1 and P2 are having the remaining burst time 7 units and 3 units respectively.

   P3 and P4 are having the remaining burst time 1 unit each. Since, both are equal hence the scheduling will be done according to their arrival time. P3 arrives earlier than P4 and therefore it will be scheduled again.

5. The Next Process P5 arrives at time unit 4. Till this time, the Process P3 has completed its execution and it is no more in the list. The scheduler will compare the remaining burst time of all the available processes. Since the burst time of process P4 is 1 which is least among all hence this will be scheduled.
6. The Next Process P6 arrives at time unit 5, till this time, the Process P4 has completed its execution. We have 4 available processes till now, that are P1 (7), P2 (3), P5 (3) and P6 (2). The Burst time of P6 is the least among all hence P6 is scheduled. Since, now, all the processes are available hence the algorithm will now work same as SJF. P6 will be executed till its completion and then the process with the least remaining time will be scheduled.

CONCLUSION :::

Shortest Job First CPU Scheduling with predicted burst time

Dynamic method – Let t_i be the actual Burst-Time of ith process and T_{n+1} be the predicted Burst-time for n+1th process.

- **Simple average –** Given n processes ($P_1, P_2 \dots P_n$)

$T_{n+1} = 1/n(\Sigma^{i=1 \text{ to } n}\ t_i)$

- **Exponential average (Aging) –**

$$T_{n+1} = \alpha t_n + (1 - \alpha)T_n$$

where α = is smoothing factor and $0 <= \alpha <= 1$,

t_n = actual burst time of nth process,
T_n = predicted burst time of nth process.

General term ::

$$\alpha t_n + (1 - \alpha)\alpha t_{n-1} + (1 - \alpha)^2\alpha t_{n-2} \dots + (1 - \alpha)^j\alpha t_{n-j} \dots + (1 - \alpha)^{n+1}T_0$$

Example –
Calculate the exponential averaging with T1 = 10, α = 0.5 and the algorithm is SJF with previous runs as 8,7, 4, 16. Which prediction of which process is coorect?
(a) 9 (b) 8 (c) 7.5 (d) None
Explanation :
Initially T1 = 10 and α = 0.5 and the run times given are 8, 7, 4, 16 as it is shortest job first,
So the possible order in which these processes would serve will be 4, 7, 8, 16 since SJF is a non-preemptive technique.
So, using formula: T2 = α*t1 + (1-α)T1
so we have,
T2 = 0.5*4 + 0.5*10 = 7, here t1 = 4 and T1 = 10
T3 = 0.5*7 + 0.5*7 = 7, here t2 = 7 and T2 = 7
T4 = 0.5*8 + 0.5*7 = 7.5, here t3 = 8 and T3 = 7
T5 = 0.5*16 + 0.5*7.5 = 11.75 here t4 = 16 and T4 = 7.5
So the future prediction for 4th process will be T4 = 7.5 which is the option(c).

The **SJF algorithm** is one of the best **scheduling** algorithms since it provides the maximum throughput and minimal waiting time but the problem with the **algorithm** is, the **CPU burst** time can't be known in advance. We can approximate the **CPU burst** time for a process.

-------------------------------- XX ---------------------------