CPU Scheduling

In the uniprogrammming systems like MS DOS, when a process waits for any I/O operation to be done, the CPU remains idol. This is an overhead since it wastes the time and causes the problem of starvation. However, In Multiprogramming systems, the CPU doesn't remain idle during the waiting time of the Process and it starts executing other processes. Operating System has to define which process the CPU will be given.

In Multiprogramming systems, the Operating system schedules the processes on the CPU to have the maximum utilization of it and this procedure is called **CPU scheduling**. The Operating System uses various scheduling algorithm to schedule the processes.

This is a task of the short term scheduler to schedule the CPU for the number of processes present in the Job Pool. Whenever the running process requests some IO operation then the short-term scheduler saves the current context of the process (also called PCB) and changes its state from running to waiting. During the time, process is in waiting state; the Short term scheduler picks another process from the ready queue and assigns the CPU to this process. This procedure is called **context switching**.

Scheduling Criteria

1. CPU Utilization

CPU utilization is a criterion used in CPU scheduling that measures the percentage of time the CPU is busy processing a task. It is important to maximize CPU utilization because when the CPU is idle, it is not performing any useful work, and this can lead to wasted system resources and reduced productivity.

A high CPU utilization indicates that the CPU is busy and working efficiently, processing as many tasks as possible. However, a too high CPU utilization can also result in a system slowdown due to excessive competition for resources.

Some CPU scheduling algorithms that prioritize CPU utilization include Round Robin, First-Come-First-Serve (FCFS), and Shortest Job First (SJF). These

algorithms aim to keep the CPU busy by assigning tasks to the CPU as soon as they are ready.

2. Throughput

Throughput is a criterion used in CPU scheduling that measures the number of tasks or processes completed within a specific period. It is important to maximize throughput because it reflects the efficiency and productivity of the system. A high throughput indicates that the system is processing tasks efficiently, which can lead to increased productivity and faster completion of tasks.

Some CPU scheduling algorithms that prioritize throughput include Round Robin, Shortest Job First (SJF), and Multilevel Queue (MLQ). These algorithms aim to prioritize short and simple tasks to increase the number of completed tasks within a specific period.

Throughput is particularly important in batch processing environments where the goal is to complete as many jobs as possible within a specific time frame. Maximizing throughput can lead to improved system performance and increased productivity.

3. Turnaround Time

Turnaround time is a criterion used in CPU scheduling that measures the time it takes for a task or process to complete from the moment it is submitted to the system until it is fully processed and ready for output. It is important to minimize turnaround time because it reflects the overall efficiency of the system and can affect user satisfaction and productivity.

A short turnaround time indicates that the system is processing tasks quickly and efficiently, leading to faster completion of tasks and improved user satisfaction. On the other hand, a long turnaround time can result in delays, decreased productivity, and user dissatisfaction.

Some CPU scheduling algorithms that prioritize turnaround time include Shortest Job First (SJF), Priority scheduling, and Multilevel Feedback Queue (MLFQ). These algorithms aim to prioritize short and simple tasks or give higher priority to more important tasks, which can lead to shorter turnaround times and improved system efficiency.

4. Waiting Time

Waiting time is a criterion used in CPU scheduling that measures the amount of time a task or process waits in the ready queue before it is processed by the CPU. It is important to minimize waiting time because it reflects the efficiency of the scheduling algorithm and affects user satisfaction.

A short waiting time indicates that tasks are being processed efficiently and quickly, leading to improved user satisfaction and productivity. On the other hand, a long waiting time can result in delays and decreased productivity, leading to user dissatisfaction. Some CPU scheduling algorithms that prioritize waiting time include Shortest Job First (SJF), Priority scheduling, and Multilevel Feedback Queue (MLFQ). These algorithms aim to prioritize short and simple tasks or give higher priority to more important tasks, which can lead to shorter waiting times and improved system efficiency.

5. Response Time

Response time is a criterion used in CPU scheduling that measures the time it takes for the system to respond to a user's request or input. It is important to minimize response time because it affects user satisfaction and the overall efficiency of the system. A short response time indicates that the system is processing tasks quickly and efficiently, leading to improved user satisfaction and productivity. On the other hand, a long response time can result in user frustration and decreased productivity. Some CPU scheduling algorithms that prioritize response time include Round Robin, Priority scheduling, and Multilevel Feedback Queue (MLFQ). These algorithms aim to prioritize tasks that require immediate attention, such as user input, to reduce response time and improve system efficiency.

What are the Types of CPU Scheduling?

There are essential 4 conditions under which CPU scheduling decisions are taken:

1. If a process is making the switch between the running state to the waiting state (could be for an I/O request, or invocation of wait() for terminating one of its child processes)
2. If a process is making the switch from the running state to the ready state (on the occurrence of an interrupt, for example)
3. If a process is making the switch between waiting and ready state (e.g. when its I/O request completes)

4. If a process terminates upon completion of execution.

So in the case of conditions 1 and 4, the CPU does not really have a choice of scheduling, if a process exists in the ready queue the CPU's response to this would be to select it for execution. In cases 2 and 3, the CPU has a choice of selecting a particular process for executing next.

There are mainly two types of CPU scheduling:

Non-Preemptive Scheduling

- Under non-preemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state.
- This scheduling method is used by the Microsoft Windows 3.1 and by the Apple Macintosh operating systems.
- It is the only method that can be used on certain hardware platforms, because It does not require the special hardware(for example: a timer) needed for preemptive scheduling.

Preemptive Scheduling

In this type of Scheduling, the tasks are usually assigned with priorities. At times it is necessary to run a certain task that has a higher priority before another task although it is running. Therefore, the running task is interrupted for some time and resumed later when the priority task has finished its execution.

Thus this type of scheduling is used mainly when a process switches either from running state to ready state or from waiting state to ready state. The resources (that is CPU cycles) are mainly allocated to the process for a limited amount of time and then are taken away, and after that, the process is again placed back in the ready queue in the case if that process still has a CPU burst time remaining. That process stays in the ready queue until it gets the next chance to execute.

Scheduling Algorithms

First Come First Serve Scheduling

In the "First come first serve" scheduling algorithm, as the name suggests, the process which arrives first, gets executed first, or we can say that the process which requests the CPU first, gets the CPU allocated first.

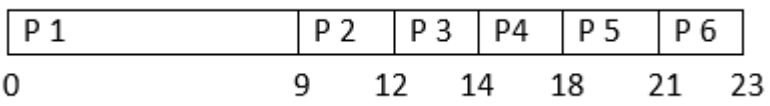
- First Come First Serve, is just like **FIFO**(First in First out) Queue data structure, where the data element which is added to the queue first, is the one who leaves the queue first.
- This is used in Batch Systems.
- It's **easy to understand and implement** programmatically, using a Queue data structure, where a new process enters through the **tail** of the queue, and the scheduler selects process from the **head** of the queue.
- A perfect real life example of FCFS scheduling is **buying tickets at ticket counter**.

Turn Around Time = Completion Time - Arrival Time

Waiting Time = Turn Around Time - Burst Time

S. No	Process ID	Arrival Time	Burst Time
1	P 1	0	9
2	P 2	1	3
3	P 3	1	2
4	P 4	1	4
5	P 5	2	3
6	P 6	3	2

Gantt chart for the above Example 1 is:



S. No	Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	P 1	0	9	9	9	0
2	P 2	1	3	12	11	8
3	P 3	1	2	14	13	11
4	P 4	1	4	18	17	13
5	P 5	2	3	21	19	16
6	P 6	3	2	23	20	18

The Average Completion Time is:

Average CT = (9 + 12 + 14 + 18 + 21 + 23) / 6

Average CT = 97 / 6

Average CT = 16.16667

The Average Waiting Time is:

Average WT = (0 + 8 + 11 + 13 + 16 + 18) /6

Average WT = 66 / 6

Average WT = 11

Shortest Job First (SJF) Scheduling

Till now, we were scheduling the processes according to their arrival time (in FCFS scheduling). However, SJF scheduling algorithm, schedules the processes according to their burst time.

In SJF scheduling, the process with the lowest burst time, among the list of available processes in the ready queue, is going to be scheduled next.

However, it is very difficult to predict the burst time needed for a process hence this algorithm is very difficult to implement in the system.

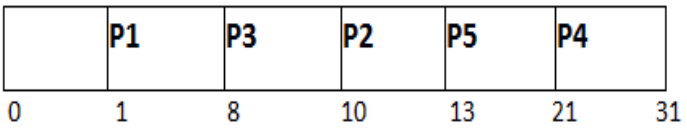
Example

In the following example, there are five jobs named as P1, P2, P3, P4 and P5. Their arrival time and burst time are given in the table below.

PID	Arrival Time	Burst Time
1	1	7

2	3	3
3	6	2
4	7	10
5	9	8

So that's how the procedure will go on in **shortest job first (SJF)** scheduling algorithm.



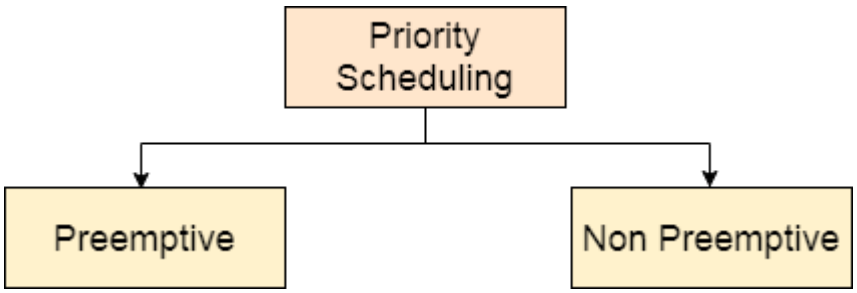
Avg Waiting Time = 27/5=5.4

AVG TAT=11.4

PID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
1	1	7	8	7	0
2	3	3	13	10	7
3	6	2	10	4	2
4	7	10	31	24	14
5	9	8	21	12	4

Priority Scheduling Algorithm in OS (Operating System)

In Priority scheduling, there is a priority number assigned to each process. In some systems, the lower the number, the higher the priority. While, in the others, the higher the number, the higher will be the priority. The Process with the higher priority among the available processes is given the CPU. There are two types of priority scheduling algorithm exists. One is **Preemptive** priority scheduling while the other is **Non Preemptive** Priority scheduling.

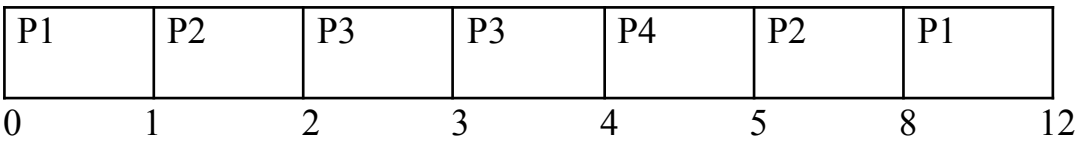


The priority number assigned to each of the process may or may not vary. If the priority number doesn't change itself throughout the process, it is called **static priority**, while if it keeps changing itself at the regular intervals, it is called **dynamic priority**.

BY using Pre-emptive Metod

PID	Arrival Time	Burst Time	Priority
1	0	5	10
2	1	4	20
3	2	2	30
4	4	1	40

So that's how the procedure will go on in the **Priority** scheduling algorithm.



Avg TAT Time = 22/4= 5.5

Avg Waiting Time = 10/4= 2.5

PID	Arrival Time	Burst Time	Priority	Completion Time	Turn Around Time	Waiting Time
1	0	5	10	12	12	7
2	1	4	20	8	7	3
3	2	2	30	4	2	0
4	4	1	40	5	1	0

Round Robin CPU Scheduling

Round Robin is a CPU scheduling algorithm where each process is cyclically assigned a fixed time slot. It is the preemptive version of the First come First Serve CPU Scheduling algorithm.

Round Robin CPU Scheduling is the most important CPU Scheduling Algorithm which is ever used in the history of CPU Scheduling Algorithms. Round Robin CPU Scheduling uses Time Quantum (TQ). The Time Quantum is

something which is removed from the Burst Time and lets the chunk of process to be completed.

Time Sharing is the main emphasis of the algorithm. Each step of this algorithm is carried out cyclically. The system defines a specific time slice, known as a time quantum.

- Round Robin CPU Algorithm generally focuses on Time Sharing technique.
- The period of time for which a process or job is allowed to run in a pre-emptive method is called time **quantum**.
- Each process or job present in the ready queue is assigned the CPU for that time quantum, if the execution of the process is completed during that time then the process will **end** else the process will go back to the **waiting table** and wait for its next turn to complete the execution

Examples:

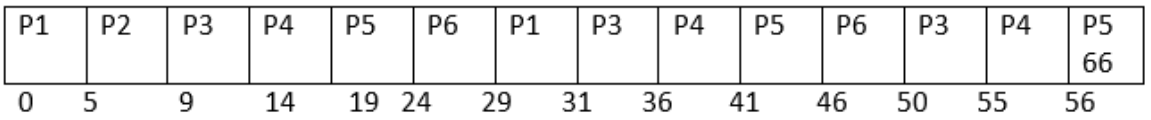
S. No	Process ID	Arrival Time	Burst Time
1	P 1	0	7
2	P 2	1	4
3	P 3	2	15
4	P 4	3	11
5	P 5	4	20
6	P 6	4	9

Assume Time Quantum TQ = 5

Ready Queue:

1. P1, P2, P3, P4, P5, P6, P1, P3, P4, P5, P6, P3, P4, P5

Gantt chart:



Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
P1	0	7	31	31	24
P2	1	4	9	8	4
P3	2	15	55	53	38
P4	3	11	56	53	42
P5	4	20	66	62	42
P6	4	9	50	46	37

Average Completion Time

1. Average Completion Time = (31 +9 + 55 +56 +66 + 50) / 6
2. Average Completion Time = 267/ 6
3. Average Completion Time = 44.5

Average Waiting Time

1. Average Waiting Time = (5 + 26 + 5 + 42 + 42 + 37) / 6
2. Average Waiting Time = 187/ 6
3. Average Waiting Time = 31.16667

Average Turn Around Time

1. Average Turn Around Time = (31 + 8 + 53 + 53 + 62 + 46) / 6
2. Average Turn Around Time = 253 / 6
3. Average Turn Around Time = 42.16667

Shortest Remaining Time Next (SRTN) Scheduling Algorithms

Selection Criteria:
The process, whose remaining run time is shortest, is served first. This is a preemptive version of SJF scheduling.

Decision Mode:
Preemptive: When a new process arrives, its total time is compared to the current process remaining run time. If the new job needs less time to finish than the current process, the current process is suspended and the new job is started.

Implementation:
This strategy can also be implemented by using sorted FIFO queue. All processes in a queue are sorted in ascending order on their remaining run time. When CPU becomes free, a process from the first position in a queue is selected to run.

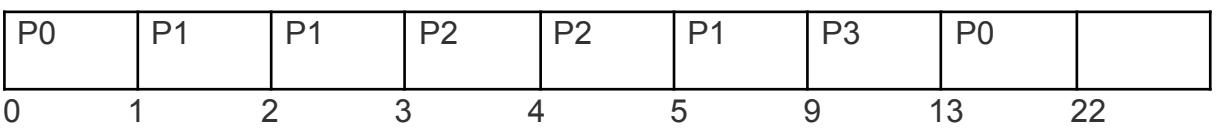
Example:
Consider the following set of four processes. Their arrival time and time required to complete the execution are given in the following table. Consider all time values in milliseconds.

Response Time : CPU first Time -AT

Process	Arrival Time (T0)	Time required for completion (ΔT)(CPU Burst Time)
P0	0	10
P1	1	6
P2	3	2
P3	5	4

Gantt Chart:

P0	P1	P2	P3	**P0**
01	3	5	13	22



Initially only process P0 is present and it is allowed to run. But, when P1 comes, it has shortest remaining run time. So, P0 is preempted and P1 is allowed to run.

Whenever new process comes or current process blocks, such type of decision is taken. This procedure is repeated till all processes complete their execution

Statistics:

Process	**Arrival Time(T0)**	**CPU Burst Time(∆T)**	**Finish Time(T1)**	**Turnaround Time(TAT=T1-T0)**	**Waiting Time (WT=TAT-∆T)**	**RT**
P0	0	10	22	22	12	0
P1	1	6	9	8	2	0
P2	3	2	5	2	0	0
P3	5	4	13	8	4	4

Average Turnaround Time: (22+8+2+8) / 4 = 40/4 = 10 ms.
Average Waiting Time: (12+2+0+4)/4 = 18 / 4 = 4.5 ms.
Advantages:

- Less waiting time.
- Quite good response for short processes.

Disadvantages:

- Again it is difficult to estimate remaining time necessary to complete execution.
- Starvation is possible for long process. Long process may wait forever.
- Context switch overhead is there.