Scheduling Criteria

Criteria of CPU Scheduling

CPU Scheduling has several criteria. Some of them are mentioned below.

1. CPU utilization

The main objective of any CPU scheduling algorithm is to keep the CPU as busy as possible. Theoretically, CPU utilization can range from 0 to 100 but in a real-time system, it varies from 40 to 90 percent depending on the load upon the system.

2. Throughput

A measure of the work done by the CPU is the number of processes being executed and completed per unit of time. This is called throughput. The throughput may vary depending on the length or duration of the processes.

3. Turnaround Time

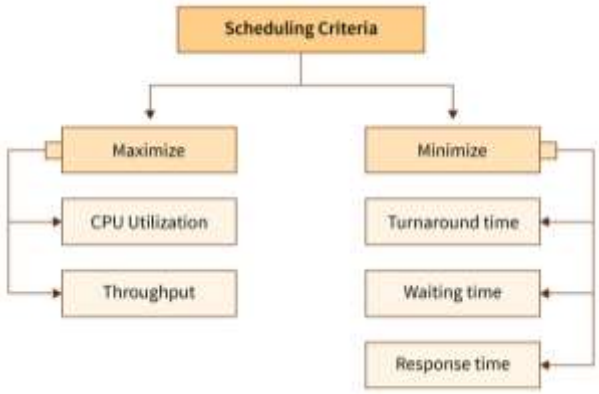
For a particular process, an important criterion is how long it takes to execute that process. The time elapsed from the time of submission of a process to the time of completion is known as the turnaround time. Turn-around time is the sum of times spent waiting to get into memory, waiting in the ready queue, executing in CPU, and waiting for I/O.

Turn Around Time = Completion Time – Arrival Time.

4. Waiting Time

A scheduling algorithm does not affect the time required to complete the process once it starts execution. It only affects the waiting time of a process i.e. time spent by a process waiting in the ready queue.

Waiting Time = Turnaround Time – Burst Time.



5. Response Time

In an interactive system, turn-around time is not the best criterion. A process may produce some output fairly early and continue computing new results while previous results are being output to the user. Thus another criterion is the time taken from submission of the process of the request until the first response is produced. This measure is called response time.

Response Time = CPU Allocation Time (when the CPU was allocated for the first) –Arrival Time

6. Completion Time

The completion time is the time when the process stops executing, which means that the process has completed its burst time and is completely executed.

7. Priority

If the operating system assigns priorities to processes, the scheduling mechanism should favor the higher-priority processes.

8. Predictability

A given process always should run in about the same amount of time under a similar system load.

Importance of Selecting the Right CPU Scheduling Algorithm for Specific Situations

It is important to choose the correct CPU scheduling algorithm because different algorithms have different priorities for different CPU scheduling criteria.Different

algorithms have different strengths and weaknesses. Choosing the wrong CPU scheduling algorithm in a given situation can result in suboptimal performance of the system.

Example: Here are some examples of CPU scheduling algorithms that work well in different situations.

Round Robin scheduling algorithm works well in a time-sharing system where tasks have to be completed in a short period of time. SJF scheduling algorithm works best in a batch processing system where shorter jobs have to be completed first in order to increase throughput. Priority scheduling algorithm works better in a real-time system where certain tasks have to be prioritized so that they can be completed in a timely manner.
Factors Influencing CPU Scheduling Algorithms

There are many factors that influence the choice of CPU scheduling algorithm. Some of them are listed below.

- The number of processes.
- The processing time required.
- The urgency of tasks.
- The system requirements.

Selecting the correct algorithm will ensure that the system will use system resources efficiently, increase productivity, and improve user satisfaction.

CPU Scheduling Algorithms

There are several CPU Scheduling Algorithms, that are listed below.

- First Come First Served (FCFS)
- Shortest Job First (SJF)
- Longest Job First (LJF)
- Priority Scheduling
- Round Robin (RR)

- Shortest Remaining Time First (SRTF)
- Longest Remaining Time First (LRTF)

FCFS Scheduling

FCFS is considered as simplest CPU-scheduling algorithm. In FCFS algorithm, the process that requests the CPU first is allocated in the CPU first. The implementation of FCFS algorithm is managed with FIFO (First in first out) queue. FCFS scheduling is non-preemptive. Nonpreemptive means, once the CPU has been allocated to a process, that process keeps the CPU until it executes a work or job or task and releases the CPU, either by requesting I/O.

Real Life Example Of FCFS Scheduling

As a real life example of FCFS scheduling a billing counter system of shopping mall can be observed. The first person in the line gets the bill done first and then the next person gets the chance to get the bill and make payment and so on.

If no priority is given to the VIP customers then the billing system will go on like this (means the first person (first task) in the line will get the bill first and after finishing (executing) the first customer's payment the counter boy(CPU) will pay attention to other customers (separate tasks) as they are in the line). As FCFS is non-preemptive type so no priority will be given to the random important tasks.

FCFS Scheduling Mathematical Examples

In CPU-scheduling problems some terms are used while solving the problems, so for conceptual purpose the terms are discussed as follows –

- **Arrival time (AT)** – Arrival time is the time at which the process arrives in ready queue.

- **Burst time (BT) or CPU time of the process** – Burst time is the unit of time in which
  a particular process completes its execution.

- **Completion time (CT)** – Completion time is the time at which the process has been terminated.

- **Turn-around time (TAT)** – The total time from arrival time to completion time is known as turn-around time. TAT can be written as,

Turn-around time (TAT) = Completion time (CT) – Arrival time (AT)
or
TAT = Burst time (BT) + Waiting time (WT)

- **Waiting time (WT)** – Waiting time is the time at which the process waits for its allocation while the previous process is in the CPU for execution. WT is written as,

Waiting time (WT) = Turn-around time (TAT) – Burst time (BT)

- **Response time (RT)** – Response time is the time at which CPU has been allocated to a particular process first time.
  In case of non-preemptive scheduling, generally Waiting time and Response time is same.

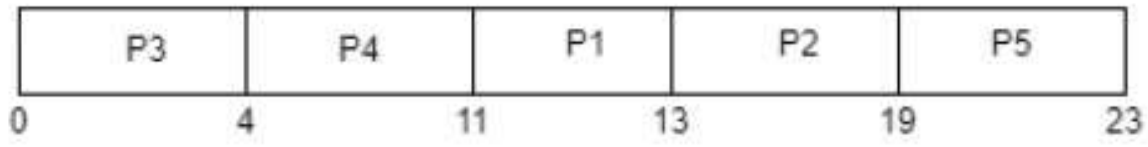
- **Gantt chart** – Gantt chart is a visualization which helps to scheduling and managing particular tasks in a project. It is used while solving scheduling problems, for a concept of how the processes are being allocated in different algorithms.

Problem 1
Consider the given table below and find Completion time (CT), Turn-around time (TAT), Waiting time (WT), Response time (RT), Average Turn-around time and Average Waiting time.

Process ID	Arrival time	Burst time
P1	2	2
P2	5	6
P3	0	4
P4	0	7
P5	7	4

Solution

Gantt chart



For this problem CT, TAT, WT, RT is shown in the given table –

Process ID	Arrival time	Burst time	CT	TAT=CT-AT	WT=TAT-BT	RT
P1	2	2	13	13-2= 11	11-2= 9	9
P2	5	6	19	19-5= 14	14-6= 8	8
P3	0	4	4	4-0= 4	4-4= 0	0
P4	0	7	11	11-0= 11	11-7= 4	4
P5	7	4	23	23-7= 16	16-4= 12	12

Average Waiting time = (9+8+0+4+12)/5 = 33/5 = 6.6 time unit (time unit can be considered as milliseconds)

Average Turn-around time = (11+14+4+11+16)/5 = 56/5 = 11.2 time unit (time unit can be considered as milliseconds)

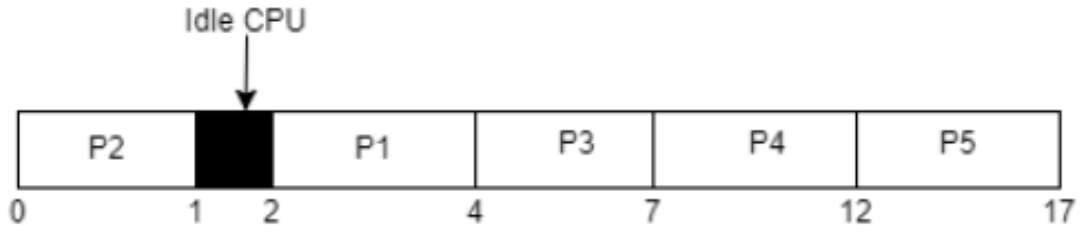
Problem 2

Consider the given table below and find Completion time (CT), Turn-around time (TAT), Waiting time (WT), Response time (RT), Average Turn-around time and Average Waiting time.

Process ID	Arrival time	Burst time
P1	2	2
P2	0	1
P3	2	3
P4	3	5
P5	4	5

Solution

Gantt chart



For this problem CT, TAT, WT, RT is shown in the given table –

Process ID	Arrival time	Burst time	CT	TAT=CT-AT	WT=TAT-BT	RT
P1	2	2	4	4-2= 2	2-2= 0	0
P2	0	1	1	1-0= 1	1-1= 0	0

P3	2	3	7	7-2= 5	5-3= 2	2
P4	3	5	12	12-3= 9	9-5= 4	4
P5	4	5	17	17-4= 13	13-5= 8	8

Average Waiting time = (0+0+2+4+8)/5 = 14/5 = 2.8 time unit (time unit can be considered as milliseconds)

Average Turn-around time = (2+1+5+9+13)/5 = 30/5 = 6 time unit (time unit can be considered as milliseconds)

*In idle (not-active) CPU period, no process is scheduled to be terminated so in this time it remains void for a little time.

Advantages Of FCFS Scheduling

- It is an easy algorithm to implement since it does not include any complex way.
- Every task should be executed simultaneously as it follows FIFO queue.
- FCFS does not give priority to any random important tasks first so it's a fair scheduling.

Disadvantages Of FCFS Scheduling

- FCFS results in convoy effect which means if a process with higher burst time comes first in the ready queue then the processes with lower burst time may get blocked and that processes with lower burst time may not be able to get the CPU if the higher burst time task takes time forever.
- If a process with long burst time comes in the line first then the other short burst time process have to wait for a long time, so it is not much good as time-sharing systems.
- Since it is non-preemptive, it does not release the CPU before it completes its task execution completely.

*Convoy effect and starvation sounds similar but there is slight difference, so it is advised to not treat these both terms as same words.

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.

Advantages of SJF

1. Maximum throughput
2. Minimum average waiting and turnaround time

Disadvantages of SJF

1. May suffer with the problem of starvation
2. It is not implementable because the exact Burst time for a process can't be known in advance.

There are different techniques available by which, the CPU burst time of the process can be determined. We will discuss them later in detail.

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	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

Since, No Process arrives at time 0 hence; there will be an empty slot in the **Gantt chart** from time 0 to 1 (the time at which the first process arrives).

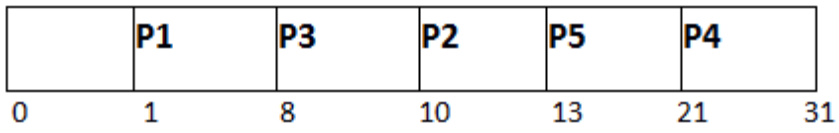
According to the algorithm, the OS schedules the process which is having the lowest burst time among the available processes in the ready queue.

Till now, we have only one process in the ready queue hence the scheduler will schedule this to the processor no matter what is its burst time.

This will be executed till 8 units of time. Till then we have three more processes arrived in the ready queue hence the scheduler will choose the process with the lowest burst time.

Among the processes given in the table, P3 will be executed next since it is having the lowest burst time among all the available processes.

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



Avg Waiting Time = 27/5

Shortest Remaining Time

- Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
- The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to give preference.

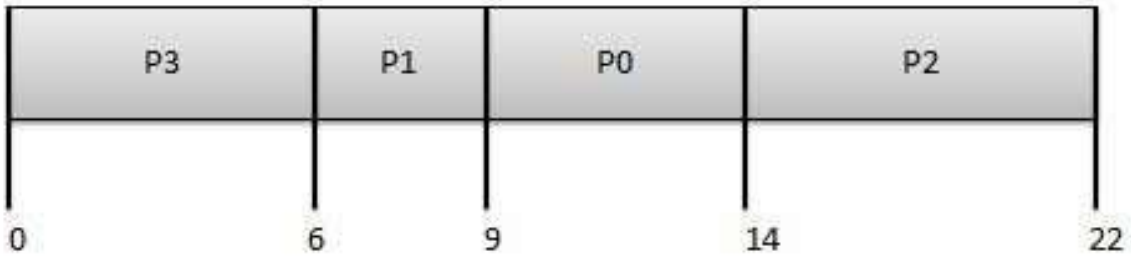
Priority Scheduling

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Processes with same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Given: Table of processes, and their Arrival time, Execution time, and priority. Here we are considering 1 is the lowest priority.

Process	Arrival Time	Execution Time	Priority	Service Time
P0	0	5	1	0
P1	1	3	2	11
P2	2	8	1	14
P3	3	6	3	5

Process	Arrival Time	Execute Time	Priority	Service Time
P0	0	5	1	9
P1	1	3	2	6
P2	2	8	1	14
P3	3	6	3	0



Waiting time of each process is as follows –

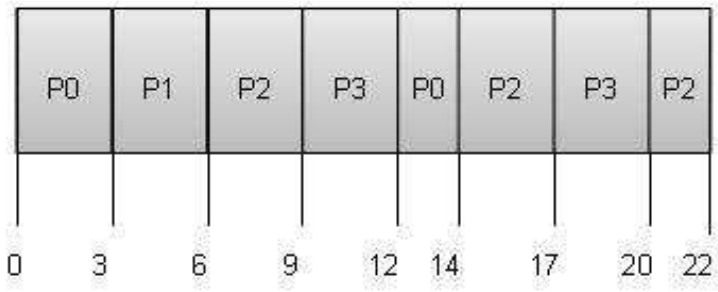
Process	**Waiting Time**
P0	0 - 0 = 0
P1	11 - 1 = 10
P2	14 - 2 = 12
P3	5 - 3 = 2

Average Wait Time: (0 + 10 + 12 + 2)/4 = 24 / 4 = 6

Round robin Scheduling

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a **quantum**.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processes.

Quantum = 3



Wait time of each process is as follows −

Process	Wait Time : Service Time - Arrival Time
P0	(0 - 0) + (12 - 3) = 9
P1	(3 - 1) = 2
P2	(6 - 2) + (14 - 9) + (20 - 17) = 12
P3	(9 - 3) + (17 - 12) = 11

Average Wait Time: (9+2+12+11) / 4 = 8.5

Multilevel Queue Scheduling

Each algorithm supports a different process, but in a general system, some processes require scheduling using a priority algorithm. While some processes want to stay in the system ***(interactive processes),*** others are ***background processes*** whose execution can be delayed.

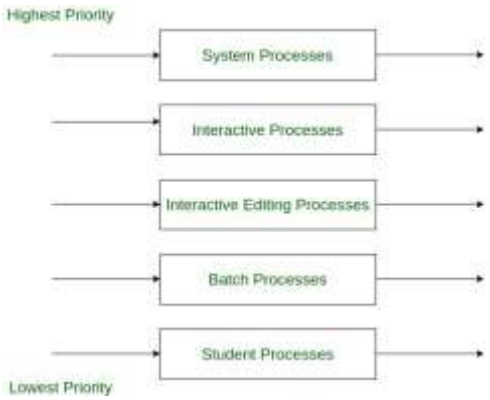
The number of ready queue algorithms between queues and within queues may differ between systems. A round-robin method with various time quantum is typically utilized for such maintenance. Several types of scheduling algorithms are designed for circumstances where the processes can be readily separated into groups. There are two sorts of processes that require different scheduling algorithms because they have

varying response times and resource requirements. ***The foreground (interactive)*** and ***background processes*** (batch process) are distinguished. Background processes take priority over foreground processes.

The ready queue has been partitioned into seven different queues using the multilevel queue scheduling technique. These processes are assigned to one queue based on their priority, such as memory size, process priority, or type. The method for scheduling each queue is different. Some queues are utilized for the foreground process, while others are used for the background process. The ***foreground queue may*** be scheduled using a ***round-robin method,*** and the ***background queue can*** be scheduled using an ***FCFS*** strategy

Example

Let's take an example of a multilevel queue-scheduling algorithm with five queues to understand how this scheduling works:



1. System process
2. Interactive processes
3. Interactive editing processes
4. Batch processes
5. Student processes

Every queue would have an absolute priority over the low-priority queues. No process may execute until the high-priority queues are empty. In the above instance, no other

process may execute until and unless the queues for system, interactive, and editing processes are empty. If an interactive editing process enters the ready queue while a batch process is underway, the batch process will be preempted.

There are the descriptions of the processes that are used in the above example:

- **System Process**
  The OS has its process to execute, which is referred to as the System Process.
- **Interactive Process**
  It is a process in which the same type of interaction should occur.
- **Batch Process**
  Batch processing is an operating system feature that collects programs and data into a batch before processing starts.
- **Student Process**
  The system process is always given the highest priority, whereas the student processes are always given the lowest.

Example Problem

Let's take an example of a multilevel queue-scheduling (MQS) algorithm that shows how the multilevel queue scheduling work. Consider the four processes listed in the table below under multilevel queue scheduling. The queue number denotes the process's queue.

Process	Arrival Time	CPU Burst Time	Queue Number
P1	0	4	1
P2	0	3	1
P3	0	8	2
P4	10	5	4

Queue 1 has a higher priority than queue **2.** Round Robin is used in queue **1 (Time Quantum = 2),** while FCFS is used in queue **2.**

Working:

1. Both queues have been processed at the start. Therefore, **queue 1 (P1, P2)** runs first (due to greater priority) in a round-robin way and finishes after 7 units.
2. The process in **queue 2 (Process P3)** starts running (since there is no process in queue 1), but while it is executing, P4 enters queue 1 and interrupts P3, and then P3 takes the CPU and finishes its execution.

Multilevel Feedback Scheduling

Multilevel Feedback Queue Scheduling (MLFQ) CPU Scheduling is like Multilevel Queue(MLQ) Scheduling but in this process can move between the queues. And thus, much more efficient than multilevel queue scheduling.

Characteristics of Multilevel Feedback Queue Scheduling:

- In a multilevel queue-scheduling algorithm, processes are permanently assigned to a queue on entry to the system, and processes are allowed to move between queues.
- As the processes are permanently assigned to the queue, this setup has the advantage of low scheduling overhead,

Features of Multilevel Feedback Queue Scheduling (MLFQ) CPU Scheduling:

Multiple queues: Similar to MLQ scheduling, MLFQ scheduling divides processes into multiple queues based on their priority levels. However, unlike MLQ scheduling, processes can move between queues based on their behavior and needs.

Priorities adjusted dynamically: The priority of a process can be adjusted dynamically based on its behavior, such as how much CPU time it has used or how often it has been blocked. Higher-priority processes are given more CPU time and lower-priority processes are given less.

Time-slicing: Each queue is assigned a time quantum or time slice, which determines how much CPU time a process in that queue is allowed to use before it is preempted and moved to a lower priority queue.

Feedback mechanism: MLFQ scheduling uses a feedback mechanism to adjust the priority of a process based on its behavior over time. For example, if a process in a lower-priority queue uses up its time slice, it may be moved to a higher-priority queue to ensure it gets more CPU time.

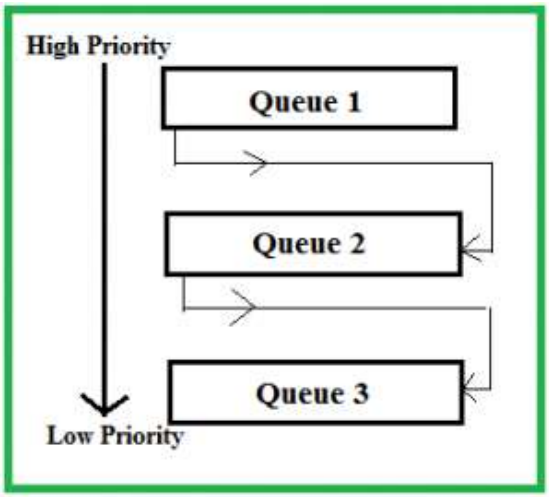
Preemption: Preemption is allowed in MLFQ scheduling, meaning that a higher-priority process can preempt a lower-priority process to ensure it gets the CPU time it needs.

Advantages of Multilevel Feedback Queue Scheduling:

- It is more flexible.
- It allows different processes to move between different queues.
- It prevents starvation by moving a process that waits too long for the lower priority queue to the higher priority queue.

Disadvantages of Multilevel Feedback Queue Scheduling:

- The selection of the best scheduler, it requires some other means to select the values.
- It produces more CPU overheads.
- It is the most complex algorithm.



Multilevel feedback queue scheduling, however, allows a process to move between queues. Multilevel Feedback Queue Scheduling **(MLFQ)** keeps analyzing the behavior (time of execution) of processes and according to which it changes its priority.
Now, look at the diagram and explanation below to understand it properly.

Now let us suppose that queues 1 and 2 follow round robin with time quantum 4 and 8 respectively and queue 3 follow FCFS.

Implementation of MFQS is given below –

- When a process starts executing the operating system can insert it into any of the above three queues depending upon its **priority**. For example, if it is some background process, then the operating system would not like it to be given to higher priority queues such as queues 1 and 2. It will directly assign it to a lower priority queue i.e. queue 3. Let's say our current process for consideration is of significant priority so it will be given **queue 1**.
- In queue 1 process executes for 4 units and if it completes in these 4 units or it gives CPU for I/O operation in these 4 units then the priority of this process does not change and if it again comes in the ready queue then it again starts its execution in Queue 1.
- If a process in queue 1 does not complete in 4 units then its priority gets reduced and it is shifted to queue 2.

- Above points 2 and 3 are also true for queue 2 processes but the time quantum is 8 units. In a general case if a process does not complete in a time quantum then it is shifted to the lower priority queue.
- In the last queue, processes are scheduled in an FCFS manner.
- A process in a lower priority queue can only execute only when higher priority queues are empty.
- A process running in the lower priority queue is interrupted by a process arriving in the higher priority queue.

Conclusion

This week we learned about the CPU Scheduling Criteria, FCFS Scheduling, SJF Scheduling, Priority Scheduling, RR Scheduling, Multilevel Queue Scheduling and Multilevel Feedback and Real-Time Scheduling.