CPU Scheduling Criteria

Different [CPU scheduling algorithms](https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/) have different properties and choice of a particular algorithm depends on the various factors. Many criteria have been suggested for comparing CPU scheduling algorithms.

The criteria include the following:

1. **CPU utilisation –**  
   The main objective of any CPU scheduling algorithm is to keep the CPU as busy as possible. Theoretically CPU utilisation 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. **Throughout –**  
   A measure of the work done by CPU is the number of processes being executed and completed per unit time. This is called throughput. The throughput may vary depending upon the length or duration of processes.
3. **Turnaround time –**  
   For a particular process, an important criteria 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 turnaround time. Turn-around time is the sum of times spent waiting to get into memory, waiting in ready queue, executing in CPU and waiting for I/O.
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.
5. **Response time –**  
   In an interactive system turn-around time is not the best criteria. A process may produce some output fairly early and continue computing new results while previous results are being output to user. Thus another criteria is the time taken from submission of process of request until the first response is produced. This measure is called response time.

**Turnaround Time (TAT):**

1. It is the time interval from the time of submission of a process to the time of the completion of the process.
2. Difference b/w Completion Time and Arrival Time is called Turnaround Time.

**Completion Time (CT):** This is the time when the process completes it’s execution.

**Formula:  
Arrival Time (AT):** This is the time when the process has arrived in the ready state./queue.

TAT = CT - AT

**Waiting Time (WT):**

1. The time spent by a process waiting in the ready queue for getting the CPU.
2. The time difference b/w Turnaround Time and Burst Time is called Waiting Time.

**Burst Time (BT):** This is the time required by the process for it’s execution.

WT = TAT - BT

Now with Waiting Time and Burst Time we can also calculate Turn Around Time via:

TAT = BT + WT

Scheduling Algorithm Optimization Criteria :

1. Max CPU utilization
2. Max throughput
3. Min turnaround time
4. Min waiting time
5. Min response time

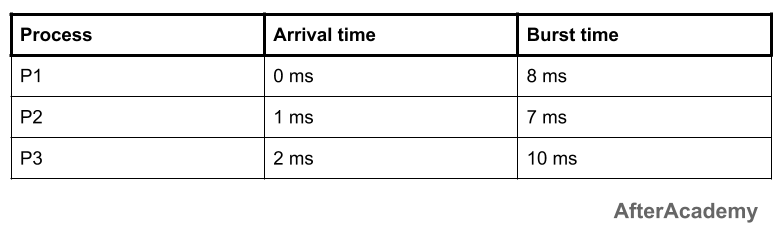
**What is Convoy Effect?(IN FCFS)**

Convoy Effect is a situation where many processes, who need to use a resource for short time are blocked by one process holding that resource for a long time.

This essentially leads to poort utilization of resources and hence poor performance.

#### Response time

Response time is the time spent when the process is in the ready state and gets the CPU for the first time. For example, here we are using the First Come First Serve CPU scheduling algorithm for the below 3 processes:



Here, the response time of all the 3 processes are:

* **P1:** 0 ms
* **P2:** 7 ms because the process P2 have to wait for 8 ms during the execution of P1 and then after it will get the CPU for the first time. Also, the arrival time of P2 is 1 ms. So, the response time will be 8-1 = 7 ms.
* **P3:** 13 ms because the process P3 have to wait for the execution of P1 and P2 i.e. after 8+7 = 15 ms, the CPU will be allocated to the process P3 for the first time. Also, the arrival of P3 is 2 ms. So, the response time for P3 will be 15-2 = 13 ms.

**Response time = Time at which the process gets the CPU for the first time - Arrival time.**

**Problem-01:**

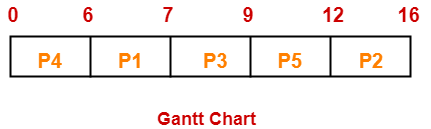
Consider the set of 5 processes whose arrival time and burst time are given below-

|  |  |  |
| --- | --- | --- |
| **Process Id** | **Arrival time** | **Burst time** |
| P1 | 3 | 1 |
| P2 | 1 | 4 |
| P3 | 4 | 2 |
| P4 | 0 | 6 |
| P5 | 2 | 3 |

If the CPU scheduling policy is SJF non-preemptive, calculate the average waiting time and average turn around time.

**Solution-**

**Gantt Chart-**



Now, we know-

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Process Id** | **Exit time/CT** | **Turn Around time** | **Waiting time** |
| P1 | 7 | 7 – 3 = 4 | 4 – 1 = 3 |
| P2 | 16 | 16 – 1 = 15 | 15 – 4 = 11 |
| P3 | 9 | 9 – 4 = 5 | 5 – 2 = 3 |
| P4 | 6 | 6 – 0 = 6 | 6 – 6 = 0 |
| P5 | 12 | 12 – 2 = 10 | 10 – 3 = 7 |

Now,

* Average Turn Around time = (4 + 15 + 5 + 6 + 10) / 5 = 40 / 5 = 8 unit
* Average waiting time = (3 + 11 + 3 + 0 + 7) / 5 = 24 / 5 = 4.8 unit

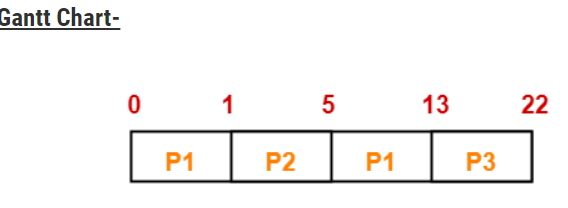
**Problem-02:**

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

|  |  |  |
| --- | --- | --- |
| **Process Id** | **Arrival time** | **Burst time** |
| P1 | 0 | 9 |
| P2 | 1 | 4 |
| P3 | 2 | 9 |

If the CPU scheduling policy is SRTF, calculate the average waiting time and average turn around time.

**Solution-**

****

Now, we know-

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Process Id** | **Exit time/CT** | **Turn Around time** | **Or,TAT=BT+WT** | **Waiting time** |
| P1 | 13 | 13 – 0 = 13 | 9+4=13 | 13 – 9 = 4 |
| P2 | 5 | 5 – 1 = 4 | 4+0=4 | 4 – 4 = 0 |
| P3 | 22 | 22- 2 = 20 | 9+11=20 | 20 – 9 = 11 |

Now,

* Average Turn Around time = (13 + 4 + 20) / 3 = 37 / 3 = 12.33 unit
* Average waiting time = (4 + 0 + 11) / 3 = 15 / 3 = 5 unit

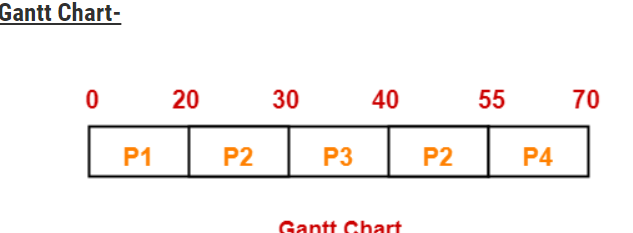
**Problem-05:**

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

|  |  |  |
| --- | --- | --- |
| **Process Id** | **Arrival time** | **Burst time** |
| P1 | 0 | 20 |
| P2 | 15 | 25 |
| P3 | 30 | 10 |
| P4 | 45 | 15 |

If the CPU scheduling policy is SRTF, calculate the waiting time of process P2.

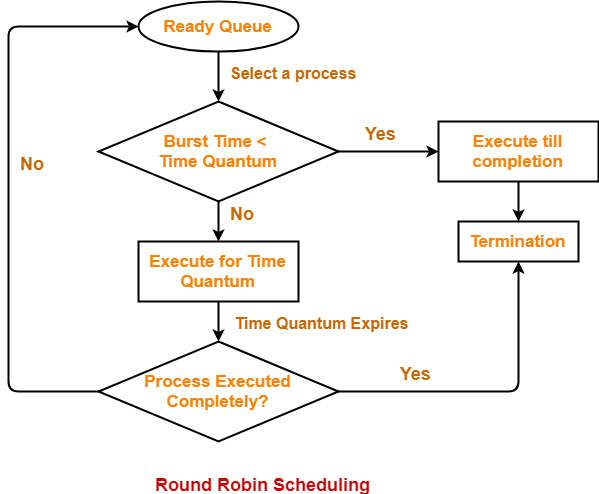
**Solution-**

****

**Rest Proceed as per formulas:-**

***ROUND ROBIN SCHEDULING***

* CPU is assigned to the process on the basis of FCFS for a fixed amount of time.
* This fixed amount of time is called as **time quantum** or **time slice**.
* After the time quantum expires, the running process is preempted and sent to the ready queue.
* Then, the processor is assigned to the next arrived process.
* It is always preemptive in nature.



## ****Advantages-****

* It gives the best performance in terms of average response time.
* It is best suited for time sharing system, client server architecture and interactive system.

## ****Disadvantages-****

* It leads to starvation for processes with larger burst time as they have to repeat the cycle many times.
* Its performance heavily depends on time quantum.
* Priorities can not be set for the processes.

## ****Important Notes-****

With decreasing value of time quantum,

* Number of context switch increases
* Response time decreases
* Chances of starvation decreases

Thus, smaller value of time quantum is better in terms of response time.

With increasing value of time quantum,

* Number of context switch decreases
* Response time increases
* Chances of starvation increases

Thus, higher value of time quantum is better in terms of number of context switch.

* With increasing value of time quantum, Round Robin Scheduling tends to become FCFS Scheduling.
* When time quantum tends to infinity, Round Robin Scheduling becomes FCFS Scheduling.

* The performance of Round Robin scheduling heavily depends on the value of time quantum.
* The value of time quantum should be such that it is neither too big nor too small.

## ****ROUND ROBIN WITH ARRIVAL TIME****

## ****Problem-01:****

Consider the set of 5 processes whose arrival time and burst time are given below-

|  |  |  |
| --- | --- | --- |
| **Process Id** | **Arrival time** | **Burst time** |
| P1 | 0 | 5 |
| P2 | 1 | 3 |
| P3 | 2 | 1 |
| P4 | 3 | 2 |
| P5 | 4 | 3 |

If the CPU scheduling policy is Round Robin with time quantum = 2 unit, calculate the average waiting time and average turn around time.

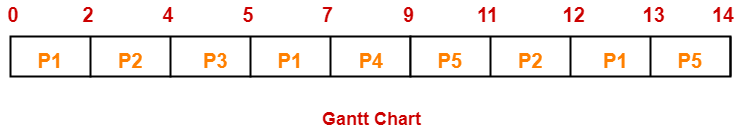
## ****Solution-****

START

### ****Gantt Chart-****

#### ****Ready Queue-****

***P5, P1, P2, P5, P4, P1, P3, P2, P1***



Now, we know-

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Process Id** | **Exit time** | **Turn Around time** | **Waiting time** |
| P1 | 13 | 13 – 0 = 13 | 13 – 5 = 8 |
| P2 | 12 | 12 – 1 = 11 | 11 – 3 = 8 |
| P3 | 5 | 5 – 2 = 3 | 3 – 1 = 2 |
| P4 | 9 | 9 – 3 = 6 | 6 – 2 = 4 |
| P5 | 14 | 14 – 4 = 10 | 10 – 3 = 7 |

Now,

* Average Turn Around time = (13 + 11 + 3 + 6 + 10) / 5 = 43 / 5 = 8.6 unit
* Average waiting time = (8 + 8 + 2 + 4 + 7) / 5 = 29 / 5 = 5.8 unit

**Multilevel queue scheduling**

Another class of scheduling algorithms has been created for situations in which processes are easily classified into different groups.

**For example:** A common division is made between foreground(or interactive) processes and background (or batch) processes. These two types of processes have different response-time requirements, and so might have different scheduling needs. In addition, foreground processes may have priority over background processes.

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:

System Processes

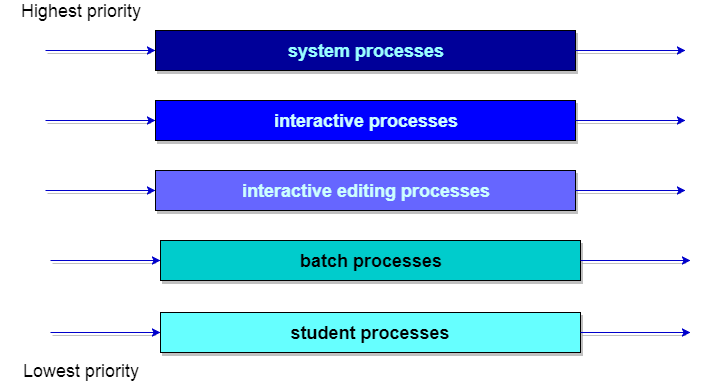
Interactive Processes

Interactive Editing Processes

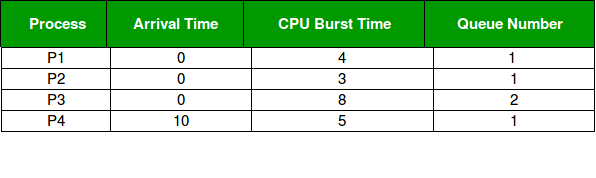
Batch Processes

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.**



**Example Problem :**   
Consider below table of four processes under Multilevel queue scheduling.Queue number denotes the queue of the process.



Priority of queue 1 is greater than queue 2. queue 1 uses Round Robin (Time Quantum = 2) and queue 2 uses FCFS.

Below is the **gantt chart** of the problem :

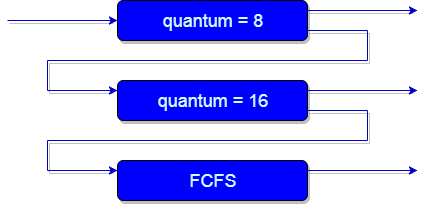
https://media.geeksforgeeks.org/wp-content/uploads/Gantt-Chart-Multilevel-Queue.png

At starting both queues have process so process in queue 1 (P1, P2) runs first (because of higher priority) in the round robin fashion and completes after 7 units then process in queue 2 (P3) starts running (as there is no process in queue 1) but while it is running P4 comes in queue 1 and interrupts P3 and start running for 5 second and after its completion P3 takes the CPU and completes its execution.

# *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.



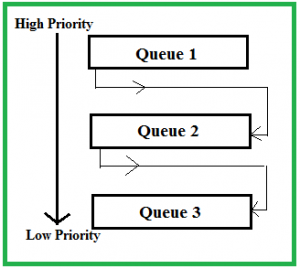
**An example of a multilevel feedback queue can be seen in the below figure.**

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.

Now, look at the diagram and explanation below to understand it properly.

EXAMPLE:-



Now let us suppose that queue 1 and 2 follow round robin with time quantum 4 and 8 respectively and queue 3 follow FCFS.One implementation of MFQS is given below –

1. When a process starts executing then it first enters queue 1.
2. In queue 1 process executes for 4 unit and if it completes in this 4 unit or it gives CPU for I/O operation in this 4 unit than the priority of this process does not change and if it again comes in the ready queue than it again starts its execution in Queue 1.
3. If a process in queue 1 does not complete in 4 unit then its priority gets reduced and it shifted to queue 2.
4. Above points 2 and 3 are also true for queue 2 processes but the time quantum is 8 unit.In a general case if a process does not complete in a time quantum than it is shifted to the lower priority queue.
5. In the last queue, processes are scheduled in FCFS manner.
6. A process in lower priority queue can only execute only when higher priority queues are empty.
7. A process running in the lower priority queue is interrupted by a process arriving in the higher priority queue.