**SCHEDULING ALGORITHMS**

CPU scheduling deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU. There are many different CPU-scheduling algorithms.

## Types of CPU scheduling Algorithm

There are mainly six types of [process scheduling algorithms](https://www.guru99.com/process-scheduling.html)

1. First Come First Serve (FCFS) Scheduling
2. Shortest-Job-First (SJF) Scheduling
3. Shortest Remaining Time First(SRTF) Scheduling
4. Highest response ratio next (HRRN) Scheduling
5. Priority Scheduling
6. Round Robin Scheduling (RR)
7. Multilevel Queue Scheduling
8. Multilevel Feedback Queue Scheduling.

**1)FIRST-COME, FIRST-SERVED SCHEDULING(FCFS):**

First-come, first-served (FCFS) scheduling algorithm is the simplest CPU scheduling algorithm. In the "First come first serve" scheduling algorithm, as the name suggests, the [process](https://www.studytonight.com/operating-system/process-scheduling) which arrives first, gets executed first, or we can say that the process which requests the CPU first, gets the CPU allocated first. In case of a tie, if two processes request CPU simultaneously, the process with a smaller process ID gets the CPU allocation first. FCFS scheduling algorithm is nonpreemptive­. Once the CPU has been allocated to a process, that process keeps the CPU until it releases the CPU, either by terminating or by requesting I/O.

**Example of FCFS:** buying tickets at the ticket counter*.*

The implementation of the FCFS policy is easily managed with a 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.

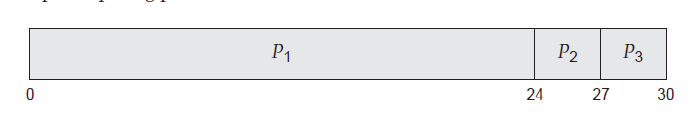
**Example:** Consider the set of 3 processes whose arrival time and burst time are given below-

|  |  |  |
| --- | --- | --- |
| Process | Arrival time | Burst time |
| P1 | 0 | 24 |
| P2 | 0 | 3 |
| P3 | 0 | 3 |

If the CPU scheduling policy is FCFS, calculate the average waiting time and average turnaround time.

**Solution:**

If the processes arrive in the order P1, P2, P3, and are served in FCFS order, we get the result shown in the following Gantt chart, which is a bar chart that illustrates a particular schedule, including the start and finish times of each of the participating processes:

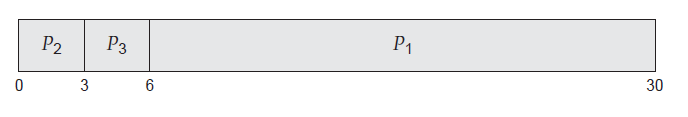


The waiting time is 0 milliseconds for process P1, 24 milliseconds for process P2, and 27 milliseconds for process P3.

Thus, the average waiting time is (0+24+27)/3 = 17 milliseconds.  
The Turnaround time is 24 milliseconds for process P1, 27 milliseconds for process P2, and 30 milliseconds for process P3.

Thus, the average Turnaround time is (24+27+30)/3 = 27 milliseconds.

On the negative side, the average waiting time under the FCFS policy is often quite long. If the processes arrive in the order P2, P3, P1, however, the results will be as shown in the following Gantt chart:

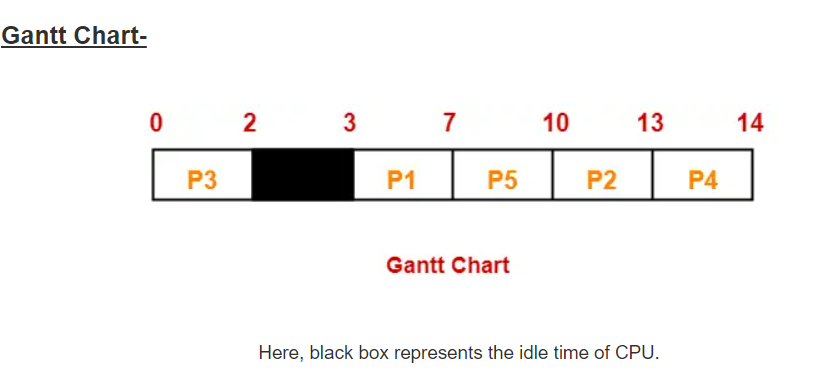


The average waiting time is now (6 + 0 + 3)/3 = 3 milliseconds. This reduction is substantial. Thus, the average waiting time under an FCFS policy is generally not minimal and may vary substantially if the processes’ CPU burst times vary greatly.

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

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

If the CPU scheduling policy is FCFS, calculate the average waiting time and average turnaround time.

****

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

Now,

Average Turn Around time = (4 + 8 + 2 + 9 + 6) / 5 = 29 / 5 = 5.8 milli seconds

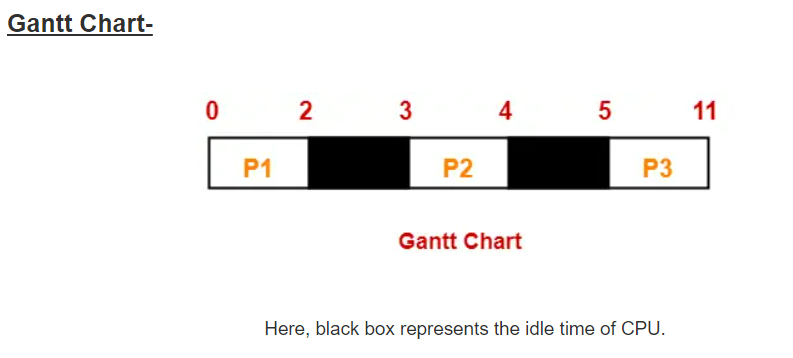
Average waiting time = (0 + 5 + 0 + 8 + 3) / 5 = 16 / 5 = 3.2 milli seconds

**Example:**

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

|  |  |  |
| --- | --- | --- |
| **Process Id** | **Arrival time** | **Burst time** |
| P1 | 0 | 2 |
| P2 | 3 | 1 |
| P3 | 5 | 6 |

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



|  |  |  |  |
| --- | --- | --- | --- |
| **Process Id** | **Exit time** | **Turn Around time** | **Waiting time** |
| P1 | 2 | 2 – 0 = 2 | 2 – 2 = 0 |
| P2 | 4 | 4 – 3 = 1 | 1 – 1 = 0 |
| P3 | 11 | 11- 5 = 6 | 6 – 6 = 0 |

Now,

* Average Turn Around time = (2 + 1 + 6) / 3 = 9 / 3 = 3 milli seconds
* Average waiting time = (0 + 0 + 0) / 3 = 0 / 3 = 0 milli seconds

**Example:**

|  |  |  |
| --- | --- | --- |
| **Process** | **Arrival time** | **Burst time** |
| P1 | 0 | 5 |
| P2 | 1 | 1 |
| P3 | 2 | 2 |
| P4 | 3 | 4 |
| P5 | 4 | 5 |
| P6 | 5 | 3 |

**Example:**

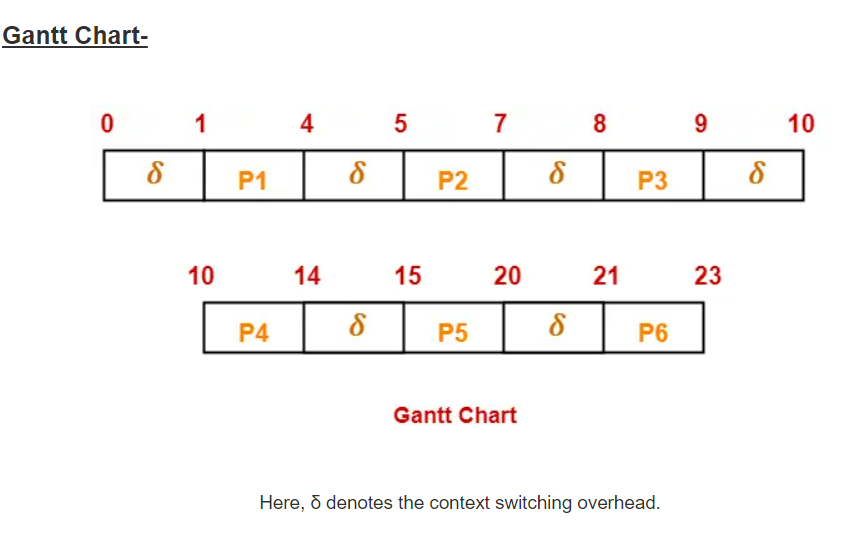
|  |  |  |
| --- | --- | --- |
| **Process** | **Arrival time** | **Burst time** |
| P1 | 5 | 4 |
| P2 | 8 | 2 |
| P3 | 6 | 3 |
| P4 | 3 | 1 |
| P5 | 2 | 2 |
| P6 | 7 | 7 |

**Example:**

Consider the set of 6 processes whose arrival time and burst time are given below-

|  |  |  |
| --- | --- | --- |
| **Process Id** | **Arrival time** | **Burst time** |
| P1 | 0 | 3 |
| P2 | 1 | 2 |
| P3 | 2 | 1 |
| P4 | 3 | 4 |
| P5 | 4 | 5 |
| P6 | 5 | 2 |

If the CPU scheduling policy is FCFS and there is 1 unit of overhead in scheduling the processes, find the efficiency of the algorithm.



Now,

* Useless time / Wasted time = 6 x δ = 6 x 1 = 6 unit
* Total time = 23 unit
* Useful time = 23 unit – 6 unit = 17 unit

Efficiency (η)= = Useful time  / Total Total

= 17 unit / 23 unit

= 0.7391

= 73.91%

**Advantages**

The advantages of the FCFS algorithm are the following:

1)This algorithm is simple to implement and easy to understand.

2) It is a fair algorithm as no priority of the processes is involved.

3) There is no starvation in the case of FCFS.

## Disadvantages

The disadvantages of the FCFS algorithm are the following:

1) This algorithm mostly suffers from the **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.

2) FCFS CPU Scheduling Algorithm has Long Waiting Time.

3) Parallel utilization of resources for processes is not possible.

4) This scheduling algorithm is not ideal for time sharing systems.