**First Come, First Serve – CPU Scheduling | (Non-preemptive)**

**FCFS Scheduling:**

Simplest [CPU scheduling algorithm](https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/) 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](https://www.geeksforgeeks.org/queue-data-structure/). When a process enters the ready queue, its [PCB](https://www.geeksforgeeks.org/process-table-and-process-control-block-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.

**Characteristics of FCFS:**

* FCFS supports non-preemptive and preemptive CPU scheduling algorithm.
* Tasks are always executed on a First-come, First-serve concept.
* FCFS is easy to implement and use.
* This algorithm is not much efficient in performance, and the wait time is quite high.

#### **How First Come First Serve CPU Scheduling Algorithm works?**

* The waiting time for first process is 0 as it is executed first.
* The waiting time for upcoming process can be calculated by:

***wt[i] =  ( at[i – 1] + bt[i – 1] + wt[i – 1] ) – at[i]***

*where*

* ***wt[i]****= waiting time of current process*
* ***at[i-1]****= arrival time of previous process*
* ***bt[i-1]****= burst time of previous process*
* ***wt[i-1]****= waiting time of previous process*
* ***at[i]****= arrival time of current process*
* The Average waiting time can be calculated by:

***Average Waiting Time = (sum of all waiting time)/(Number of processes)***

#### **Examples to show working of Non-Preemptive First come first serve CPU Scheduling Algorithm:**

**Example-1:**Consider the following table of [arrival time and burst time](https://www.geeksforgeeks.org/difference-between-arrival-time-and-burst-time-in-cpu-scheduling/) for five processes **P1, P2, P3, P4** and **P5**.

| Processes | Arrival Time | Burst Time |
| --- | --- | --- |
| P1 | 0 | 4 |
| P2 | 1 | 3 |
| P3 | 2 | 1 |
| P4 | 3 | 2 |
| P5 | 4 | 5 |

The**First come First serve** CPU Scheduling Algorithm will work on the basis of steps as mentioned below:

**Step 0:**At time = 0,

* The process begins with **P1**
* As it has arrival time 0

| Time Instance | Process | Arrival Time | Waiting Table | Execution Time | Initial Burst Time | Remaining Burst  Time |
| --- | --- | --- | --- | --- | --- | --- |
| 0-1ms | P1 | 0ms |  | 1ms | 4ms | 3ms |

**Step 1:**At time = 1,

* The process **P2**arrives
* But process **P1**still executing,
* Thus, **P2**kept in a waiting table and wait for its execution.

| Time Instance | Process | Arrival Time | Waiting Table | Execution Time | Initial Burst Time | Remaining Burst  Time |
| --- | --- | --- | --- | --- | --- | --- |
| 1-2ms | P1 | 0ms |  | 1ms | 3ms | 2ms |
| P2 | 1ms | P2 | 0ms | 3ms | 3ms |

**Step 3:**At time = 2,

* The process **P3**arrives and kept in a waiting queue
* While process **P1**is still executing as its burst time is 4.

| Time Instance | Process | Arrival Time | Waiting Table | Execution Time | Initial Burst Time | Remaining Burst  Time |
| --- | --- | --- | --- | --- | --- | --- |
| 2-3ms | P1 | 0ms |  | 1ms | 2ms | 1ms |
| P2 | 1ms | P2 | 0ms | 3ms | 3ms |
| P3 | 2ms | P2, P3 | 0ms | 1ms | 1ms |

**Step 4:**At time = 3,

* The process **P4**arrives and kept in the waiting queue
* While process **P1**is still executing as its burst time is **4**

| Time Instance | Process | Arrival Time | Waiting Table | Execution Time | Initial Burst Time | Remaining Burst  Time |
| --- | --- | --- | --- | --- | --- | --- |
| 3-4ms | ~~P1~~ | ~~0ms~~ |  | ~~1ms~~ | ~~1ms~~ | ~~0ms~~ |
| P2 | 1ms | P2 | 0ms | 3ms | 3ms |
| P3 | 2ms | P2, P3 | 0ms | 1ms | 1ms |
| P4 | 3ms | P2, P3, P4 | 0ms | 2ms | 2ms |

**Step 5:** At time = 4,

* The process **P1**completes its execution
* Process **P5**arrives in waiting queue while process **P2**starts executing

| Time Instance | Process | Arrival Time | Waiting Table | Execution Time | Initial Burst Time | Remaining Burst  Time |
| --- | --- | --- | --- | --- | --- | --- |
| 4-5ms | P2 | 1ms |  | 1ms | 3ms | 2ms |
| P3 | 2ms | P3 | 0ms | 1ms | 1ms |
| P4 | 3ms | P3, P4 | 0ms | 2ms | 2ms |
| P5 | 4ms | P3, P4, P5 | 0ms | 5ms | 5ms |

**Step 6:**At time = 5,

* The process **P2**completes its execution

| Time Instance | Process | Arrival Time | Waiting Table | Execution Time | Initial Burst Time | Remaining Burst  Time |
| --- | --- | --- | --- | --- | --- | --- |
| 5-7ms | ~~P2~~ | ~~1ms~~ |  | ~~2ms~~ | ~~2ms~~ | ~~0ms~~ |
| P3 | 2ms | P3 | 0ms | 1ms | 1ms |
| P4 | 3ms | P3, P4 | 0ms | 2ms | 2ms |
| P5 | 4ms | P3, P4, P5 | 0ms | 5ms | 5ms |

**Step 7:**At time = 7,

* Process **P3**starts executing, it has burst time of 1 thus, it completes execution at time interval 8

| Time Instance | Process | Arrival Time | Waiting Table | Execution Time | Initial Burst Time | Remaining Burst  Time |
| --- | --- | --- | --- | --- | --- | --- |
| 7-8ms | ~~P3~~ | ~~2ms~~ |  | ~~1ms~~ | ~~1ms~~ | ~~0ms~~ |
| P4 | 3ms | P4 | 0ms | 2ms | 2ms |
| P5 | 4ms | P4, P5 | 0ms | 5ms | 5ms |

**Step 8:**At time 8,

* The process of **P3**completes its execution
* Process **P4**starts executing, it has burst time of 2 thus, it completes execution at time interval 10.

| Time Instance | Process | Arrival Time | Waiting Table | Execution Time | Initial Burst Time | Remaining Burst  Time |
| --- | --- | --- | --- | --- | --- | --- |
| 8-10ms | ~~P4~~ | ~~3ms~~ |  | ~~2ms~~ | ~~2ms~~ | ~~0ms~~ |
| P5 | 4ms | P5 | 0ms | 5ms | 5ms |

**Step 9:**At time 10,

* The process **P4**completes its execution
* Process **P5**starts executing, it has burst time of 5 thus, it completes execution at time interval 15.

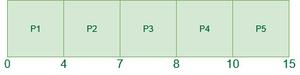
| Time Instance | Process | Arrival Time | Waiting Table | Execution Time | Initial Burst Time | Remaining Burst  Time |
| --- | --- | --- | --- | --- | --- | --- |
| 10-15ms | ~~P5~~ | ~~4ms~~ |  | ~~5ms~~ | ~~5ms~~ | ~~0ms~~ |

**Step 10:**At time 15,

* Process P5 will finish its execution.
* The overall execution of the processes will be as shown below:

| Time Instance | Process | Arrival Time | Waiting Table | Execution Time | Initial Burst Time | Remaining Burst  Time |
| --- | --- | --- | --- | --- | --- | --- |
| 0-1ms | P1 | 0ms |  | 1ms | 4ms | 3ms |
| 1-2ms | P1 | 0ms |  | 1ms | 3ms | 2ms |
| P2 | 1ms | P2 | 0ms | 3ms | 3ms |
| 2-3ms | P1 | 0ms |  | 1ms | 2ms | 1ms |
| P2 | 1ms | P2 | 0ms | 3ms | 3ms |
| P3 | 2ms | P2, P3 | 0ms | 1ms | 1ms |
| 3-4ms | **~~P1~~** | **~~0ms~~** |  | **~~1ms~~** | **~~1ms~~** | **~~0ms~~** |
| P2 | 1ms | P2 | 0ms | 3ms | 3ms |
| P3 | 2ms | P2, P3 | 0ms | 1ms | 1ms |
| P4 | 3ms | P2, P3, P4 | 0ms | 2ms | 2ms |
| 4-5ms | P2 | 1ms |  | 1ms | 3ms | 2ms |
| P3 | 2ms | P3 | 0ms | 1ms | 1ms |
| P4 | 3ms | P3, P4 | 0ms | 2ms | 2ms |
| P5 | 4ms | P3, P4, P5 | 0ms | 5ms | 5ms |
| 5-7ms | **~~P2~~** | **~~1ms~~** |  | **~~2ms~~** | **~~2ms~~** | **~~0ms~~** |
| P3 | 2ms | P3 | 0ms | 1ms | 1ms |
| P4 | 3ms | P3, P4 | 0ms | 2ms | 2ms |
| P5 | 4ms | P3, P4, P5 | 0ms | 5ms | 5ms |
| 7-8ms | **~~P3~~** | **~~2ms~~** |  | **~~1ms~~** | **~~1ms~~** | **~~0ms~~** |
| P4 | 3ms | P4 | 0ms | 2ms | 2ms |
| P5 | 4ms | P4, P5 | 0ms | 5ms | 5ms |
| 8-10ms | **~~P4~~** | **~~3ms~~** |  | **~~2ms~~** | **~~2ms~~** | **~~0ms~~** |
| P5 | 4ms | P5 | 0ms | 5ms | 5ms |
| 10-15ms | **~~P5~~** | **~~4ms~~** |  | **~~5ms~~** | **~~5ms~~** | **~~0ms~~** |

**Gantt chart for above execution:**



*Gantt chart for First come First serve Scheduling*

***Waiting Time = Start time – Arrival time***

**P1**= 0 – 0 = 0  
**P2**= 4 – 1 = 3  
**P3**= 7 – 2 = 5  
**P4**= 8 – 3 = 5  
**P5**= 10 – 4 = 6

***Average waiting time****= 0 + 3 + 5 + 5+ 6 / 5 = 19 / 5 =****3.8***

**Complexity Analysis:**

* **Time Complexity:** O(N)
* **Auxiliary Space:**O(N)

**Advantages of FCFS:**

* The simplest and basic form of CPU Scheduling algorithm
* Easy to implement
* First come first serve method

***Disadvantages of FCFS:***

* As it is a Non-preemptive CPU Scheduling Algorithm, hence it will run till it finishes the execution.
* Average waiting time in the FCFS is much higher than the others
* It suffers from Convoy effect.
* Not very efficient due to its simplicity
* Processes which are at the end of the queue, have to wait longer to finish.