|  |  |  |
| --- | --- | --- |
| **CL-2006 Operating Systems** | **LAB - 06**  **FCFS CPU scheduling algorithm, SJF CPU scheduling algorithm** | |
|  | |  |

**First Come First Serve – CPU Scheduling**

Simplest [CPU scheduling algorithm](https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/) that schedules according to arrival times of processes. The 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 to 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 algorithms.
* Tasks are always executed on a First-come, First-serve concept.
* FCFS is easy to implement and use.
* This algorithm is not very efficient in performance, and the wait time is quite high.

**Algorithm for FCFS Scheduling**

* The waiting time for the first process is 0 as it is executed first.
* The waiting time for the 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)***

[***Preemptive Priority Scheduling Algorithm***](https://www.geeksforgeeks.org/difference-between-preemptive-priority-based-and-non-preemptive-priority-based-cpu-scheduling-algorithms/)***:****In Preemptive Priority Scheduling Algorithm, the processes come with a priority attached to them. The lower the priority number, the higher is the priority attached to the process. A process with higher priority on its arrival preempts an ongoing process. It gets the CPU. The process with priority 1 always gets the CPU whenever it arrives and is never preempted. It has a response time of 0. Equal priority processes are scheduled in the FCFS order.*

***Example –****Here is a list of processes, their arrival time, and burst time. The Gantt Chart shows how they are executed.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***PROCESS*** | ***ARRIVAL TIME*** | ***BURST TIME*** | | ***PRIORITY*** |
| ***TOTAL*** | ***REMAINING*** |
| ***P1*** | ***0*** | ***4*** | ***4*** | ***4*** |
| ***P2*** | ***1*** | ***3*** | ***3*** | ***3*** |
| ***P3*** | ***3*** | ***4*** | ***4*** | ***1*** |
| ***P4*** | ***6*** | ***2*** | ***2*** | ***5*** |
| ***P5*** | ***8*** | ***4*** | ***4*** | ***2*** |

*A green rectangular object with black text

Description automatically generated*

Task 01:

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 |

***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***

***Task 02:***

***Now add priority 1,4,0,3,2 to the matrix and solve it via preemptive approach and then calculate the waiting and average waiting time.***

**Shortest Job First (SJF)** is an algorithm in which the process having the smallest execution time is chosen for the next execution. This scheduling method can be preemptive or non-preemptive. It significantly reduces the average waiting time for other processes awaiting execution. The full form of SJF is Shortest Job First.

**There are basically two types of SJF methods:**

* Non-Preemptive SJF
* Preemptive SJF

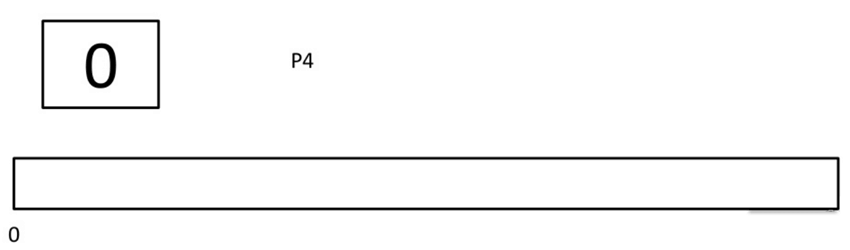
## Non-Preemptive SJF

In non-preemptive scheduling, once the CPU cycle is allocated to process, the process holds it till it reaches a waiting state or terminated.

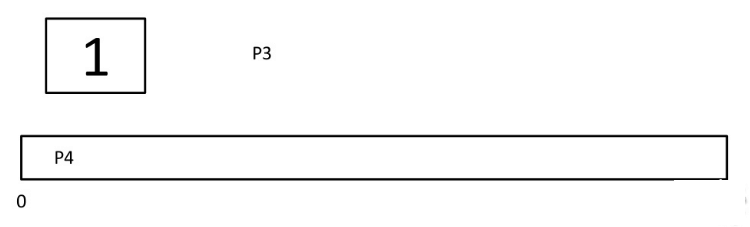
Consider the following five processes each having its own unique burst time and arrival time.

| **Process Queue** | **Burst time** | **Arrival time** |
| --- | --- | --- |
| P1 | 6 | 2 |
| P2 | 2 | 5 |
| P3 | 8 | 1 |
| P4 | 3 | 0 |
| P5 | 4 | 4 |

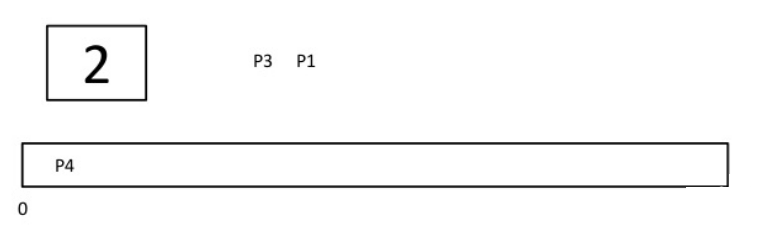
**Step 0)**At time=0, P4 arrives and starts execution.

[](https://www.guru99.com/images/1/122419_0538_ShortestJob1.png)

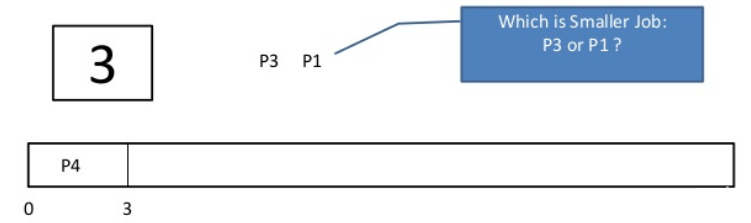
**Step 1)**At time= 1, Process P3 arrives. But, P4 still needs 2 execution units to complete. It will continue execution.

[](https://www.guru99.com/images/1/122419_0538_ShortestJob2.png)

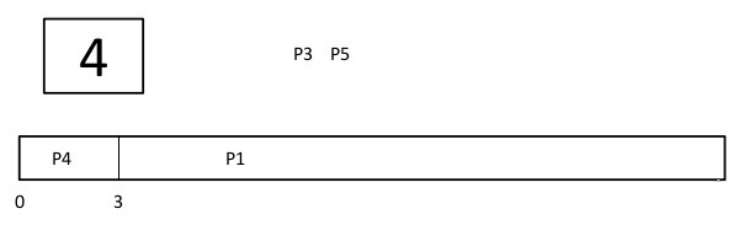
**Step 2)**At time =2, process P1 arrives and is added to the waiting queue. P4 will continue execution.

[](https://www.guru99.com/images/1/122419_0538_ShortestJob3.png)

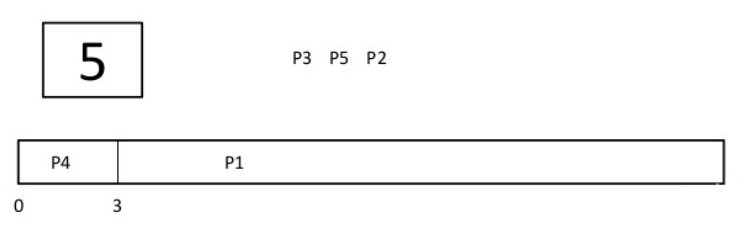
**Step 3)**At time = 3, process P4 will finish its execution. The burst time of P3 and P1 is compared. Process P1 is executed because its burst time is less compared to P3.

[](https://www.guru99.com/images/1/122419_0538_ShortestJob4.png)

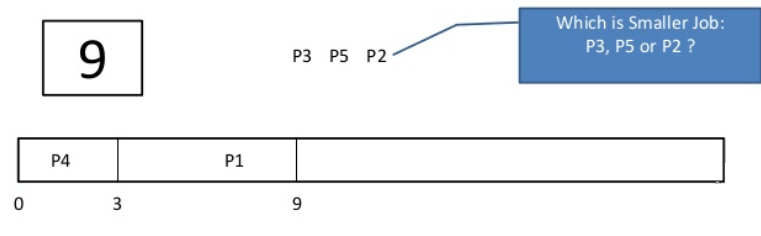
**Step 4)**At time = 4, process P5 arrives and is added to the waiting queue. P1 will continue execution.

[](https://www.guru99.com/images/1/122419_0538_ShortestJob5.png)

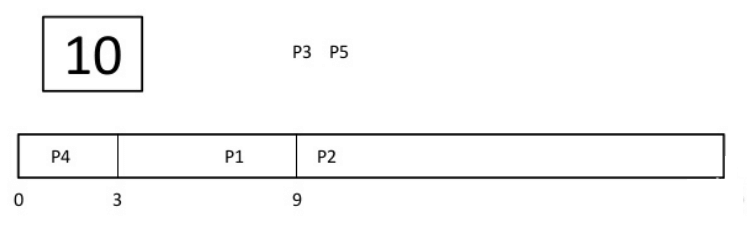
**Step 5)** At time = 5, process P2 arrives and is added to the waiting queue. P1 will continue execution.

[](https://www.guru99.com/images/1/122419_0538_ShortestJob6.png)

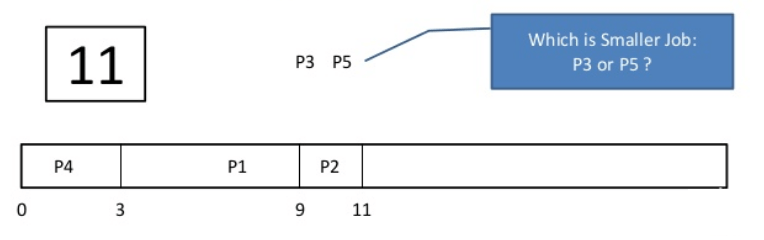
**Step 6)**At time = 9, process P1 will finish its execution. The burst time of P3, P5, and P2 is compared. Process P2 is executed because its burst time is the lowest.

[](https://www.guru99.com/images/1/122419_0538_ShortestJob7.png)

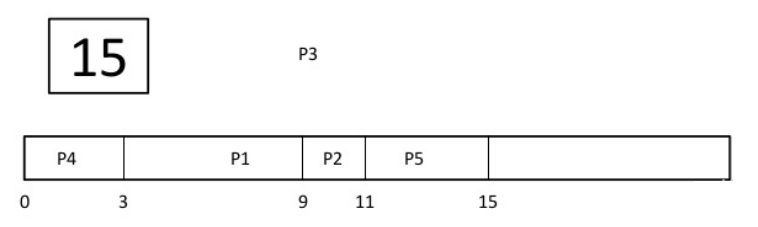
**Step 7)**At time=10, P2 is executing and P3 and P5 are in the waiting queue.

[](https://www.guru99.com/images/1/122419_0538_ShortestJob8.png)

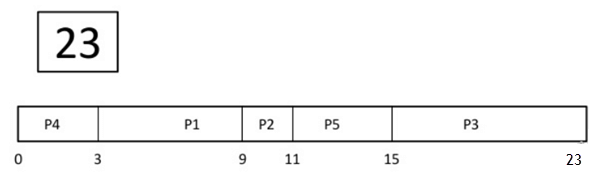
**Step 8)**At time = 11, process P2 will finish its execution. The burst time of P3 and P5 is compared. Process P5 is executed because its burst time is lower.

[](https://www.guru99.com/images/1/122419_0538_ShortestJob9.png)

**Step 9)**At time = 15, process P5 will finish its execution.

[](https://www.guru99.com/images/1/122419_0538_ShortestJob10.png)

**Step 10)**At time = 23, process P3 will finish its execution.

[](https://www.guru99.com/images/1/122419_0538_ShortestJob11.png)

**Step 11)**Let’s calculate the average waiting time for above example.

Wait time

P4= 0-0=0

P1= 3-2=1

P2= 9-5=4

P5= 11-4=7

P3= 15-1=14

Average Waiting Time= 0+1+4+7+14/5 = 26/5 = 5.2

## Preemptive SJF

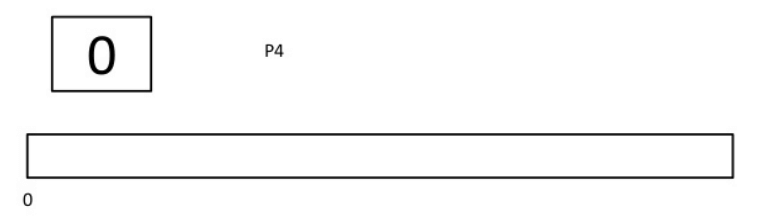
In Preemptive SJF Scheduling, jobs are put into the ready queue as they come. A process with shortest burst time begins execution. If a process with even a shorter burst time arrives, the current process is removed or preempted from execution, and the shorter job is allocated CPU cycle.

Consider the following five process:

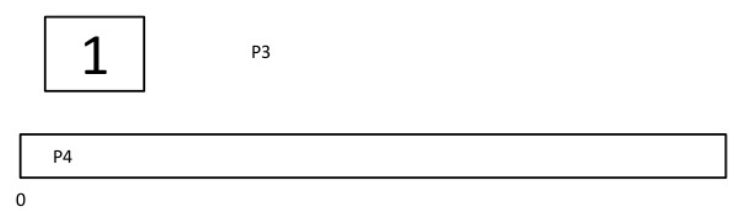
| **Process Queue** | **Burst time** | **Arrival time** |
| --- | --- | --- |
| P1 | 6 | 2 |
| P2 | 2 | 5 |
| P3 | 8 | 1 |
| P4 | 3 | 0 |
| P5 | 4 | 4 |

**Step 0)**At time=0, P4 arrives and starts execution.

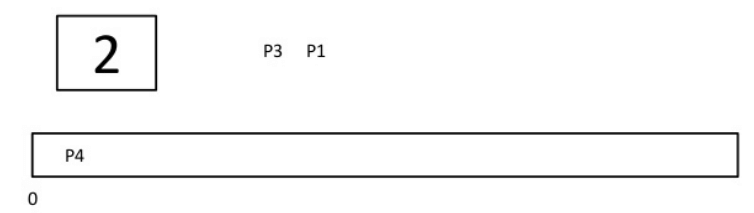
| **Process Queue** | **Burst time** | **Arrival time** |
| --- | --- | --- |
| P1 | 6 | 2 |
| P2 | 2 | 5 |
| P3 | 8 | 1 |
| P4 | 3 | 0 |
| P5 | 4 | 4 |

[](https://www.guru99.com/images/1/122419_0538_ShortestJob12.png)

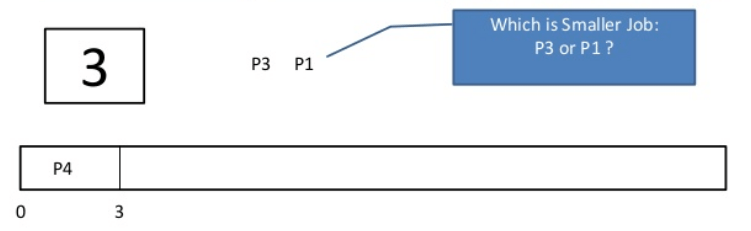
**Step 1)**At time= 1, Process P3 arrives. But, P4 has a shorter burst time. It will continue execution.

[](https://www.guru99.com/images/1/122419_0538_ShortestJob13.png)

**Step 2)**At time = 2, process P1 arrives with burst time = 6. The burst time is more than that of P4. Hence, P4 will continue execution.

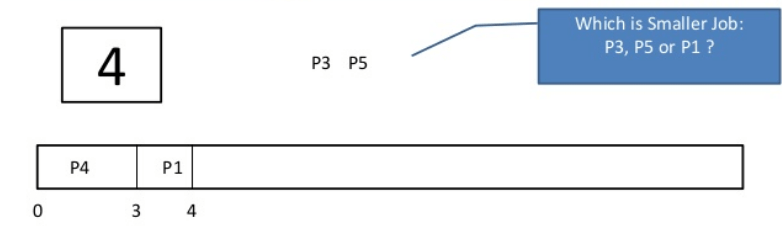
[](https://www.guru99.com/images/1/122419_0538_ShortestJob14.png)

**Step 3)**At time = 3, process P4 will finish its execution. The burst time of P3 and P1 is compared. Process P1 is executed because its burst time is lower.

[](https://www.guru99.com/images/1/122419_0538_ShortestJob15.png)

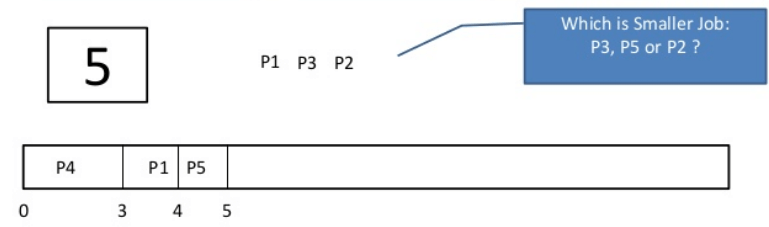
**Step 4)**At time = 4, process P5 will arrive. The burst time of P3, P5, and P1 is compared. Process P5 is executed because its burst time is lowest. Process P1 is preempted.

| **Process Queue** | **Burst time** | **Arrival time** |
| --- | --- | --- |
| P1 | 5 out of 6 is remaining | 2 |
| P2 | 2 | 5 |
| P3 | 8 | 1 |
| P4 | 3 | 0 |
| P5 | 4 | 4 |

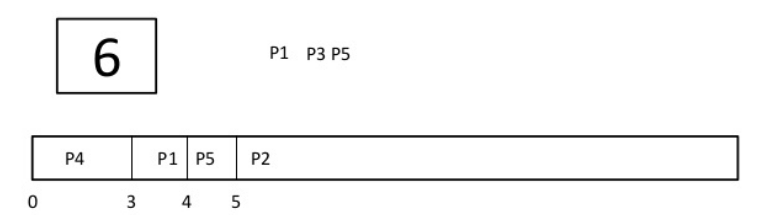
[](https://www.guru99.com/images/1/122419_0538_ShortestJob16.png)

**Step 5)**At time = 5, process P2 will arrive. The burst time of P1, P2, P3, and P5 is compared. Process P2 is executed because its burst time is least. Process P5 is preempted.

| **Process Queue** | **Burst time** | **Arrival time** |
| --- | --- | --- |
| P1 | 5 out of 6 is remaining | 2 |
| P2 | 2 | 5 |
| P3 | 8 | 1 |
| P4 | 3 | 0 |
| P5 | 3 out of 4 is remaining | 4 |

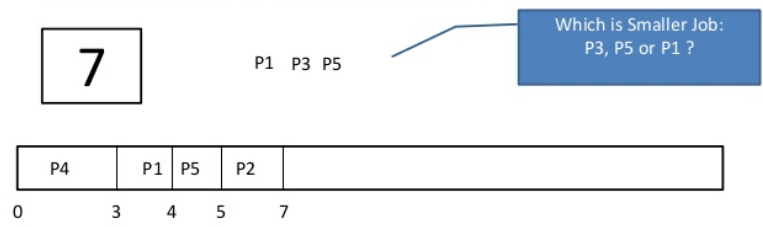
[](https://www.guru99.com/images/1/122419_0538_ShortestJob17.png)

**Step 6)**At time =6, P2 is executing.

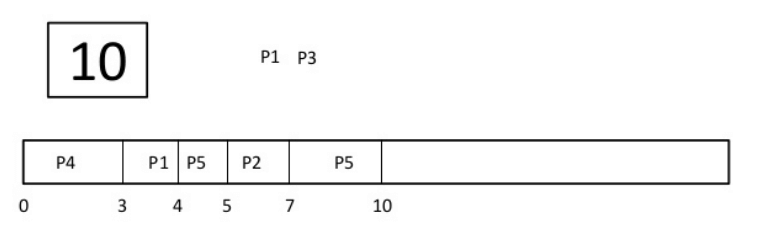
[](https://www.guru99.com/images/1/122419_0538_ShortestJob18.png)

**Step 7)**At time =7, P2 finishes its execution. The burst time of P1, P3, and P5 is compared. Process P5 is executed because its burst time is lesser.

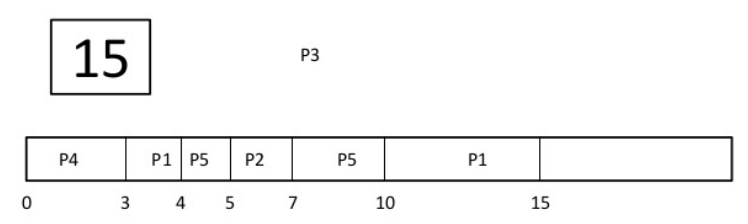
| **Process Queue** | **Burst time** | **Arrival time** |
| --- | --- | --- |
| P1 | 5 out of 6 is remaining | 2 |
| P2 | 2 | 5 |
| P3 | 8 | 1 |
| P4 | 3 | 0 |
| P5 | 3 out of 4 is remaining | 4 |

[](https://www.guru99.com/images/1/122419_0538_ShortestJob19.png)

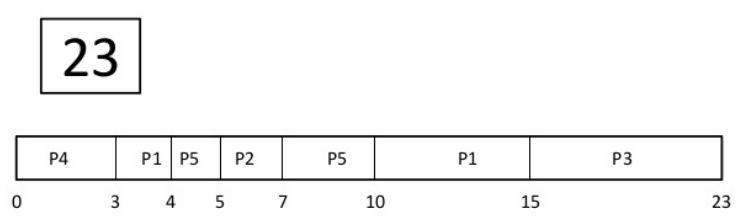
**Step 8)** At time =10, P5 will finish its execution. The burst time of P1 and P3 is compared. Process P1 is executed because its burst time is less.

[](https://www.guru99.com/images/1/122419_0538_ShortestJob20.png)

**Step 9)**At time =15, P1 finishes its execution. P3 is the only process left. It will start execution.

[](https://www.guru99.com/images/1/122419_0538_ShortestJob21.png)

**Step 10)**At time =23, P3 finishes its execution.

[](https://www.guru99.com/images/1/122419_0538_ShortestJob22.png)

**Step 11)**Let’s calculate the average waiting time for above example.  
Wait time   
P4= 0-0=0  
P1= (3-2) + 6 =7  
P2= 5-5 = 0  
P5= 4-4+2 =2  
P3= 15-1 = 14  
Average Waiting Time = 0+7+0+2+14/5 = 23/5 =4.6

***Task 03: Attempt the above task via code in either C or C++ for both non-preemptive and preemptive   
Task 04: Create a table of your own and run both algorithms side-by-side and compute the difference between their average waiting times and execution times.***