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

At time=0, P4 arrives and starts execution.

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

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

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.

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

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

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.

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

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.

|  |  |  |
| --- | --- | --- |
| **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 |

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

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

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.

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 |

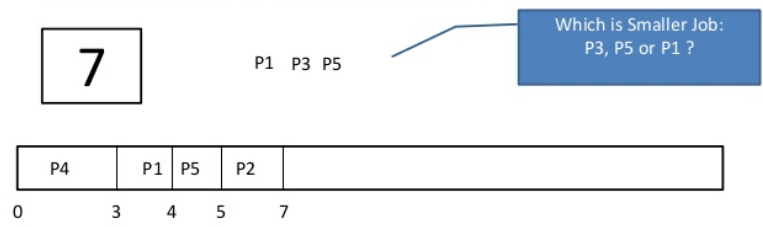
At time = 5, process P2 will arrive. The burst time of P1, P2, P3, and P5 is compared. Process P2 is executed because its burst time is least. Process P5 is preempted.

|  |  |  |
| --- | --- | --- |
| **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 |

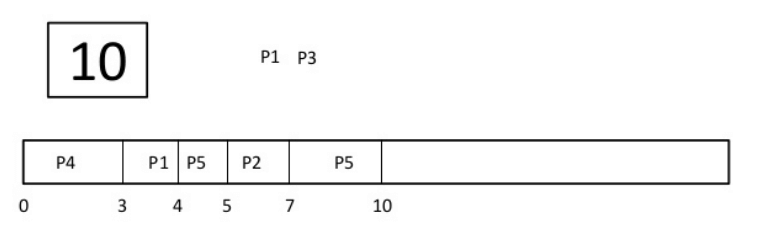
At time =6, P2 is executing.

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.

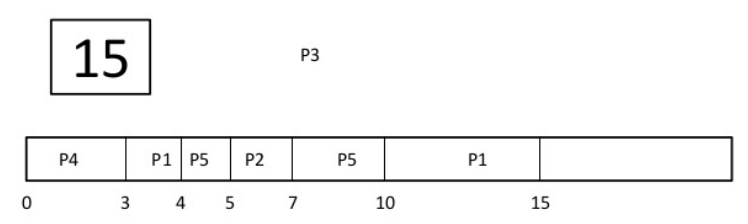
|  |  |  |
| --- | --- | --- |
| **Process Queue** | **Burst time** | **Arrival time** |
| P1 | 5 out of 6 is remaining | 2 |
| P2 | 2 | 5 |
| P3 | 8 | 1 |
| P4 | 3 | 0 |
| P5 | 3 out of 4 is remaining | 4 |



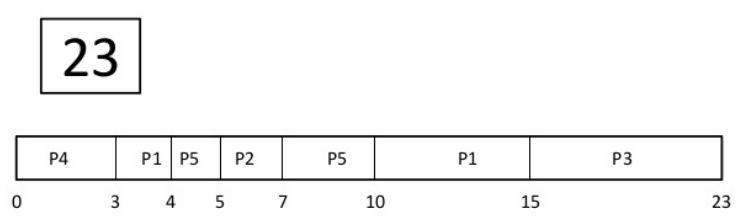
At time =10, P5 will finish its execution. The burst time of P1 and P3 is compared. Process P1 is executed because its burst time is less.



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



At time =23, P3 finishes its execution.



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

**Advantages of SJF**

Here are the benefits/pros of using SJF method:

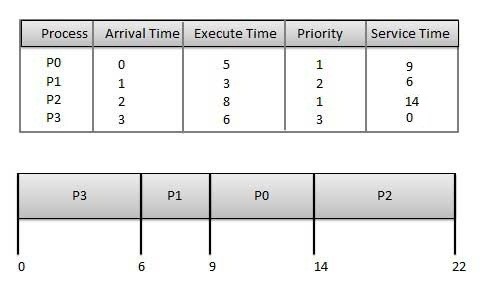
* SJF is frequently used for long term scheduling.
* It reduces the average waiting time over FIFO (First in First Out) algorithm.
* SJF method gives the lowest average waiting time for a specific set of processes.
* It is appropriate for the jobs running in batch, where run times are known in advance.
* For the batch system of long-term scheduling, a burst time estimate can be obtained from the job description.
* For Short-Term Scheduling, we need to predict the value of the next burst time.
* Probably optimal with regard to average turnaround time.

**Disadvantages/Cons of SJF**

Here are some drawbacks/cons of SJF algorithm:

* Job completion time must be known earlier, but it is hard to predict.
* It is often used in a batch system for long term scheduling.
* SJF can’t be implemented for CPU scheduling for the short term. It is because there is no specific method to predict the length of the upcoming CPU burst.
* This algorithm may cause very long turnaround times or starvation.
* Requires knowledge of how long a process or job will run.
* It leads to the starvation that does not reduce average turnaround time.
* It is hard to know the length of the upcoming CPU request.
* Elapsed time should be recorded, that results in more overhead on the processor.
* 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 |



**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 |

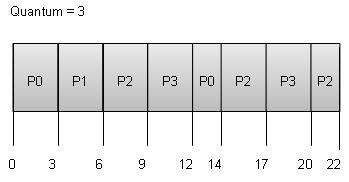
Average Wait Time: (0 + 10 + 12 + 2)/4 = 24 / 4 = 6

## 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.

## 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



**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