**Student Name=** Vinay Kumar

**Section=** K18GA

**Project no. =** 14

**Reg no.=** 11813144

**Roll no. =** B64

**Email Address=** vk9014891@gmail.com

**GitHub Link=**

**What is Priority Scheduling?**

**Priority Scheduling** is a method of scheduling processes that is based on priority. In this algorithm, the scheduler selects the tasks to work as per the priority.

The processes with higher priority should be carried out first, whereas jobs with equal priorities are carried out on a round-robin or FCFS basis. Priority depends upon memory requirements, time requirements, etc

Processes with the same priority are executed on first come first served basis. Priority can be decided based on memory requirements, time requirements or any other resource requirement.

## Types of Priority Scheduling

Priority scheduling divided into two main types:

### Preemptive Scheduling

In Preemptive Scheduling, the tasks are mostly assigned with their priorities. Sometimes it is important to run a task with a higher priority before another lower priority task, even if the lower priority task is still running. The lower priority task holds for some time and resumes when the higher priority task finishes its execution.

### Non-Preemptive Scheduling

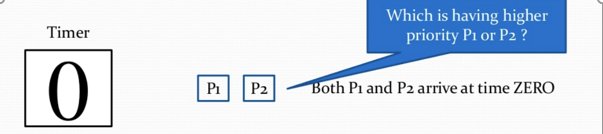
In this type of scheduling method, the CPU has been allocated to a specific process. The process that keeps the CPU busy, will release the CPU either by switching context or terminating. It is the only method that can be used for various hardware platforms. That's because it doesn't need special hardware (for example, a timer) like preemptive scheduling.

**Example of Priority Scheduling OR Algorithm**

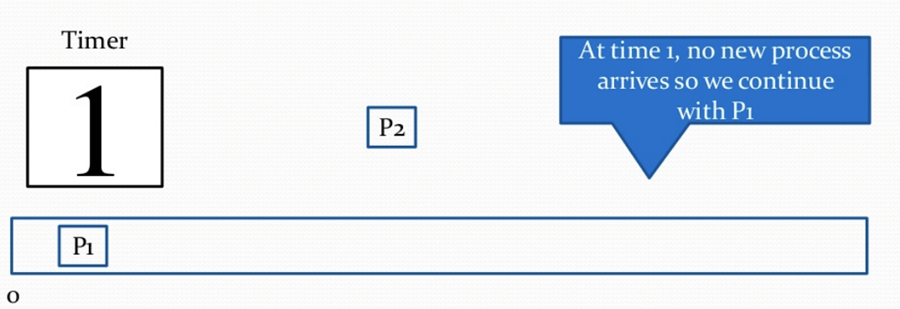
Consider following five processes P1 to P5. Each process has its unique priority, burst time, and arrival time.

| **Process** | **Priority** | **Burst time** | **Arrival time** |
| --- | --- | --- | --- |
| P1 | 1 | 4 | 0 |
| P2 | 2 | 3 | 0 |
| P3 | 1 | 7 | 6 |
| P4 | 3 | 4 | 11 |
| P5 | 2 | 2 | 12 |

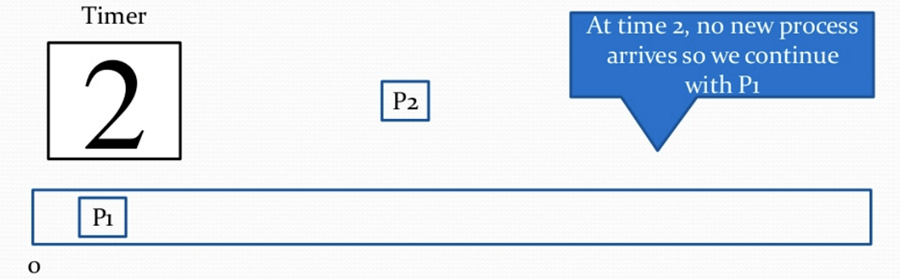
**Step 0)** At time=0, Process P1 and P2 arrive. P1 has higher priority than P2. The execution begins with process P1, which has burst time 4.

[](https://www.guru99.com/images/1/122419_0509_PrioritySch1.png)

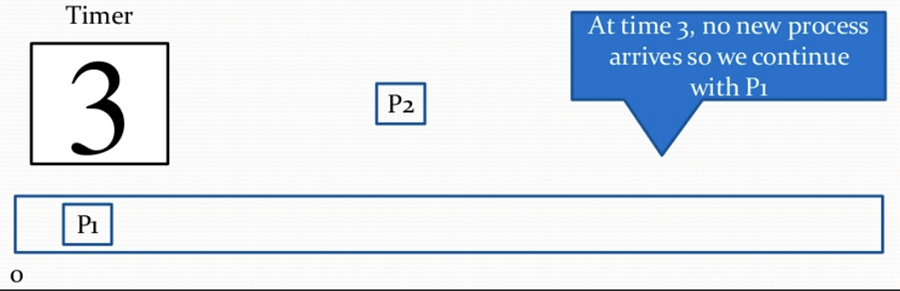
**Step 1)** At time=1, no new process arrive. Execution continues with P1.

[](https://www.guru99.com/images/1/122419_0509_PrioritySch2.png)

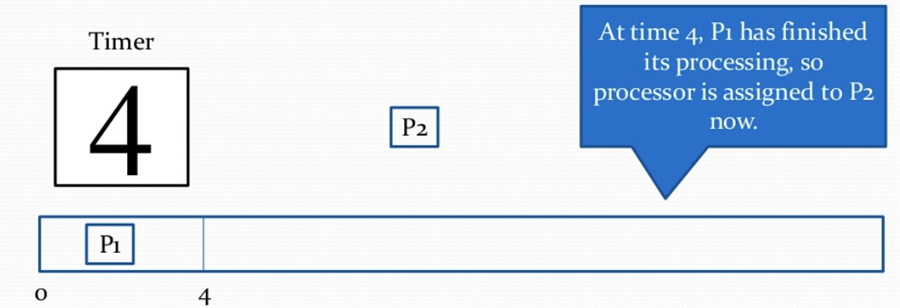
**Step 2)**At time 2, no new process arrives, so you can continue with P1. P2 is in the waiting queue.

[](https://www.guru99.com/images/1/122419_0509_PrioritySch3.png)

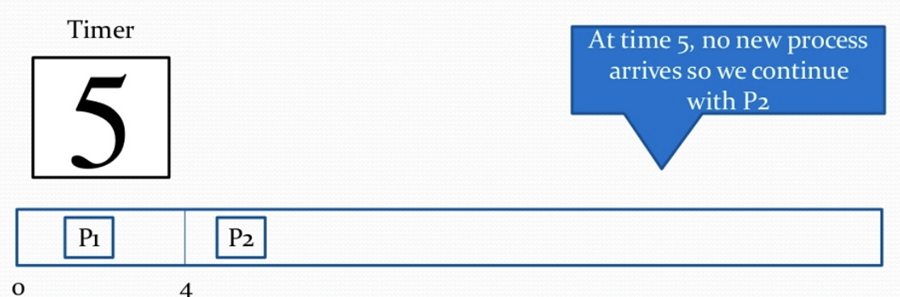
**Step 3)** At time 3, no new process arrives so you can continue with P1. P2 process still in the waiting queue.

[](https://www.guru99.com/images/1/122419_0509_PrioritySch4.png)

**Step 4)** At time 4, P1 has finished its execution. P2 starts execution.

[](https://www.guru99.com/images/1/122419_0509_PrioritySch5.png)

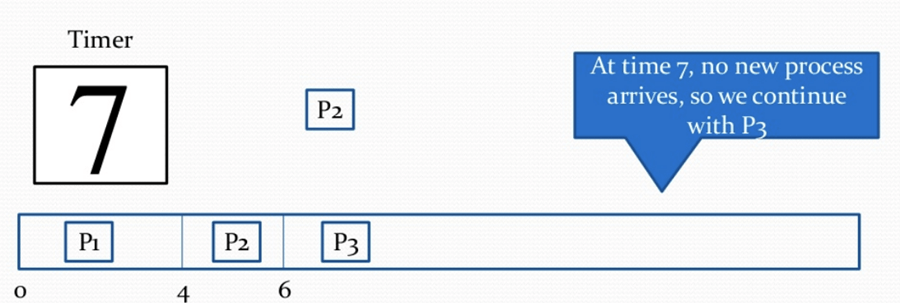
**Step 5)**At time= 5, no new process arrives, so we continue with P2.

[](https://www.guru99.com/images/1/122419_0509_PrioritySch6.png)

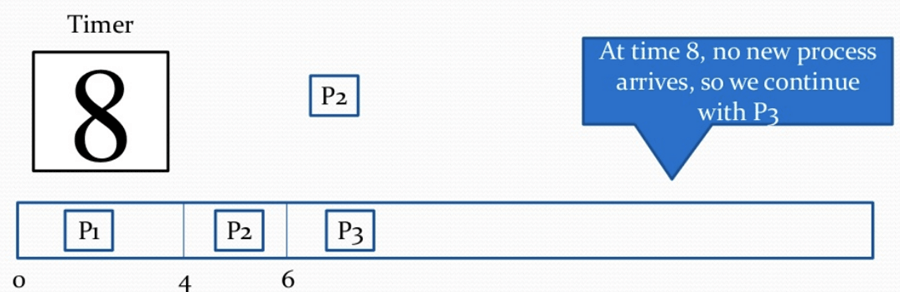
**Step 6)**At time=6, P3 arrives. P3 is at higher priority (1) compared to P2 having priority (2). P2 is preempted, and P3 begins its execution.

| **Process** | **Priority** | **Burst time** | **Arrival time** |
| --- | --- | --- | --- |
| P1 | 1 | 4 | 0 |
| P2 | 2 | 1 out of 3 pending | 0 |
| P3 | 1 | 7 | 6 |
| P4 | 3 | 4 | 11 |
| P5 | 2 | 2 | 12 |

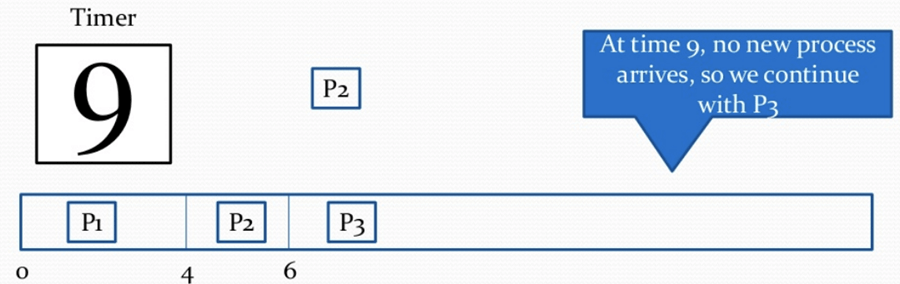
**Step 7) At** time 7, no-new process arrives, so we continue with P3. P2 is in the waiting queue.

[](https://www.guru99.com/images/1/122419_0509_PrioritySch8.png)

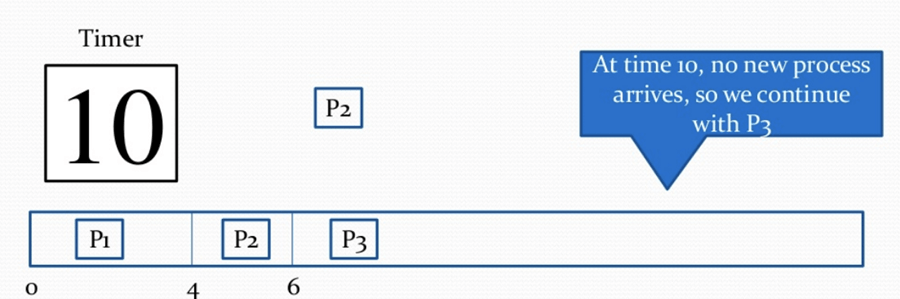
**Step 8)**At time= 8, no new process arrives, so we can continue with P3.

[](https://www.guru99.com/images/1/122419_0509_PrioritySch9.png)

**Step 9)** At time= 9, no new process comes so we can continue with P3.

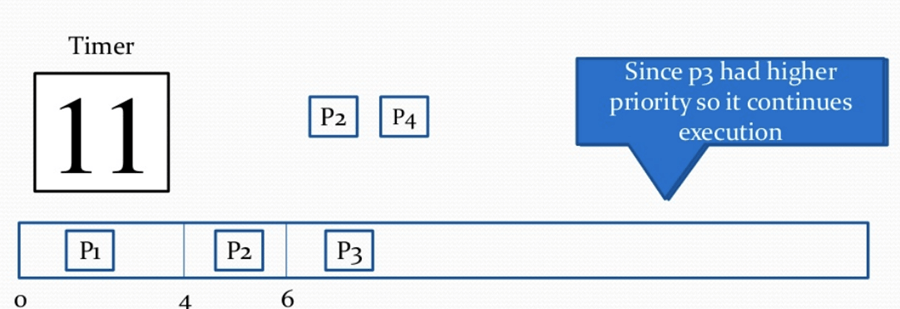
[](https://www.guru99.com/images/1/122419_0509_PrioritySch10.png)

**Step 10)**At time interval 10, no new process comes, so we continue with P3

[](https://www.guru99.com/images/1/122419_0509_PrioritySch11.png)

**Step 11)** At time=11, P4 arrives with priority 4. P3 has higher priority, so it continues its execution.

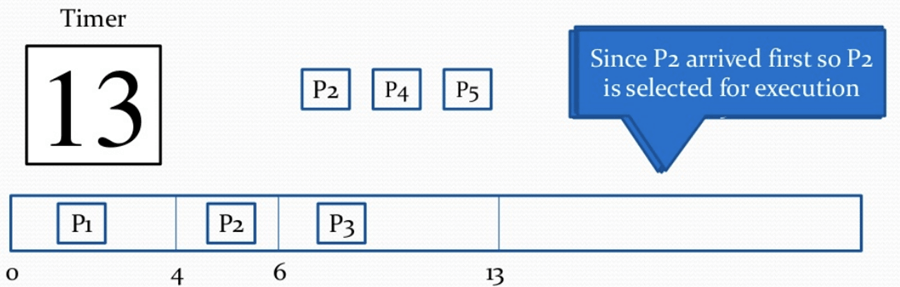
| **Process** | **Priority** | **Burst time** | **Arrival time** |
| --- | --- | --- | --- |
| P1 | 1 | 4 | 0 |
| P2 | 2 | 1 out of 3 pending | 0 |
| P3 | 1 | 2 out of 7 pending | 6 |
| P4 | 3 | 4 | 11 |
| P5 | 2 | 2 | 12 |

[](https://www.guru99.com/images/1/122419_0509_PrioritySch12.png)

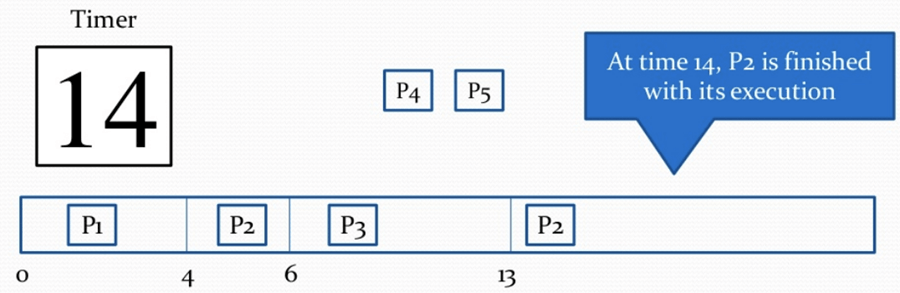
**Step 12)**At time=12, P5 arrives. P3has higher priority, so it continues execution.

**Step 13)**At time=13, P3 completes execution. We haveP2,P4,P5 in ready queue. P2 and P5 have equal priority. Arrival time of P2 is before P5. So P2 starts execution.

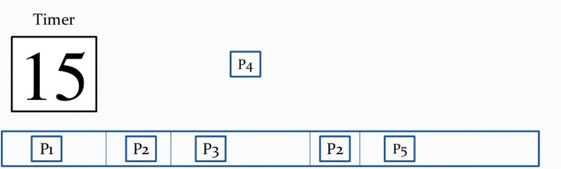
| **Process** | **Priority** | **Burst time** | **Arrival time** |
| --- | --- | --- | --- |
| P1 | 1 | 4 | 0 |
| P2 | 2 | 1 out of 3 pending | 0 |
| P3 | 1 | 7 | 6 |
| P4 | 3 | 4 | 11 |
| P5 | 2 | 2 | 12 |

[](https://www.guru99.com/images/1/122419_0509_PrioritySch14.png)

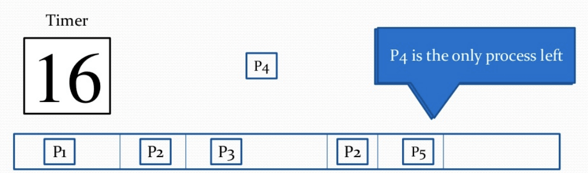
**Step 14)**At time =14, the P2 process has finished its execution. P4 and P5 are in the waiting state. P5 has the highest priority and starts execution.

[](https://www.guru99.com/images/1/122419_0509_PrioritySch15.png)

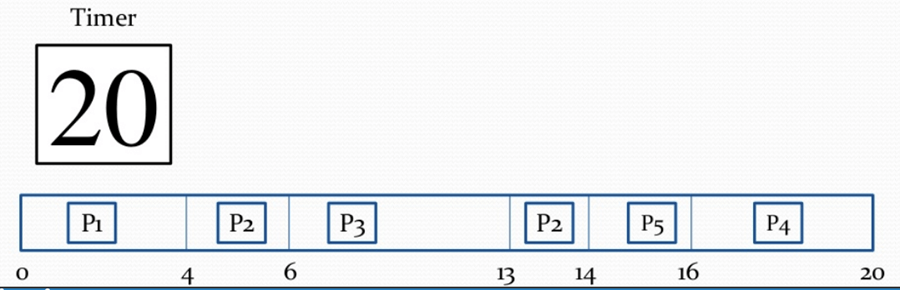
**Step 15)**At time =15, P5 continues execution.

[](https://www.guru99.com/images/1/122419_0509_PrioritySch16.png)

**Step 16)** At time= 16, P5 is finished with its execution. P4 is the only process left. It starts execution.

[](https://www.guru99.com/images/1/122419_0509_PrioritySch17.png)

**Step 17)** At time =20, P5 has completed execution and no process is left.

[](https://www.guru99.com/images/1/122419_0509_PrioritySch18.png)

**Step 18)** Let's calculate the average waiting time for the above example.

Waiting Time = start time - arrival time + wait time for next burst

P1 = o - o = o

P2 =4 - o + 7 =11

P3= 6-6=0

P4= 16-11=5

Average Waiting time = (0+11+0+5+2)/5 = 18/5= 3.6

## Advantages of priority scheduling

Here, are benefits/pros of using priority scheduling method:

* Easy to use scheduling method
* Processes are executed on the basis of priority so high priority does not need to wait for long which saves time
* This method provides a good mechanism where the relative important of each process may be precisely defined.
* Suitable for applications with fluctuating time and resource requirements.

## Disadvantages of priority scheduling

Here, are cons/drawbacks of priority scheduling

* If the system eventually crashes, all low priority processes get lost.
* If high priority processes take lots of CPU time, then the lower priority processes may starve and will be postponed for an indefinite time.
* This scheduling algorithm may leave some low priority processes waiting indefinitely.
* A process will be blocked when it is ready to run but has to wait for the CPU because some other process is running currently.
* If a new higher priority process keeps on coming in the ready queue, then the process which is in the waiting state may need to wait for a long duration of time.

**Code:**

#include<stdio.h>

int main()

{

int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg\_wt,avg\_tat;

printf("Enter Total Number of Process:");

scanf("%d",&n);

printf("\nEnter Burst Time and Priority\n");

for(i=0;i<n;i++)

{

printf("\nP[%d]\n",i+1);

printf("Burst Time:");

scanf("%d",&bt[i]);

printf("Priority:");

scanf("%d",&pr[i]);

p[i]=i+1; //contains process number

}

//sorting burst time, priority and process number in ascending order using selection sort

for(i=0;i<n;i++)

{

pos=i;

for(j=i+1;j<n;j++)

{

if(pr[j]<pr[pos])

pos=j;

}

temp=pr[i];

pr[i]=pr[pos];

pr[pos]=temp;

temp=bt[i];

bt[i]=bt[pos];

bt[pos]=temp;

temp=p[i];

p[i]=p[pos];

p[pos]=temp;

}

wt[0]=0; //waiting time for first process is zero

//calculate waiting time

for(i=1;i<n;i++)

{

wt[i]=0;

for(j=0;j<i;j++)

wt[i]+=bt[j];

total+=wt[i];

}

avg\_wt=total/n; //average waiting time

total=0;

printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");

for(i=0;i<n;i++)

{

tat[i]=bt[i]+wt[i]; //calculate turnaround time

total+=tat[i];

printf("\nP[%d]\t\t %d\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);

}

avg\_tat=total/n; //average turnaround time

printf("\n\nAverage Waiting Time=%d",avg\_wt);

printf("\nAverage Turnaround Time=%d\n",avg\_tat);

return 0;

}

**Code output:**

