Priority Scheduling

Priority Scheduling

↳ i) Process with highest priority is executed first.

ii) Processes with same priority are executed in FCFS manner.

iii) Problem of Starvation may occur.

Example. Calculate Average Waiting time for the following:-

Process	Burst Time	Priority
P_1	21	2
P_2	3	1
P_3	6	4
P_4	2	3

Assume all Process arrives at 0.

Types of Priority Scheduling Algorithm

Priority scheduling can be of two types:

1. **Preemptive Priority Scheduling**: If the new process arrived at the ready queue has a higher priority than the currently running process, the CPU is preempted, which means the processing of the current process is stoped and the incoming new process with higher priority gets the CPU for its execution.
2. **Non-Preemptive Priority Scheduling**: In case of non-preemptive priority scheduling algorithm if a new process arrives with a higher priority than the current running process, the incoming process is put at the head of the ready queue, which means after the execution of the current process it will be processed.

Problem with Priority Scheduling Algorithm

In priority scheduling algorithm, the chances of **indefinite blocking** or **starvation**.

A process is considered blocked when it is ready to run but has to wait for the CPU as some other process is running currently.

But in case of priority scheduling if new higher priority processes keeps coming in the ready queue then the processes waiting in the ready queue with lower priority may have to wait for long durations before getting the CPU for execution.

> In 1973, when the IBM 7904 machine was shut down at MIT, a low-priority process was found which was submitted in 1967 and had not yet been run.

Characteristics of Priority Scheduling

- A CPU algorithm that schedules processes based on priority.
- It used in Operating systems for performing batch processes.
- If two jobs having the same priority are READY, it works on a FIRST COME, FIRST SERVED basis.
- In priority scheduling, a number is assigned to each process that indicates its priority level.
- Lower the number, higher is the priority.
- In this type of scheduling algorithm, if a newer process arrives, that is having a higher priority than the currently running process, then the currently running process is preempted.

Example of Priority Scheduling (Non-Preemptive)

FCFS manner.

$P_4 \rightarrow$ Aging (increases priority of process)

iiii) Problem of Starvation may occur.

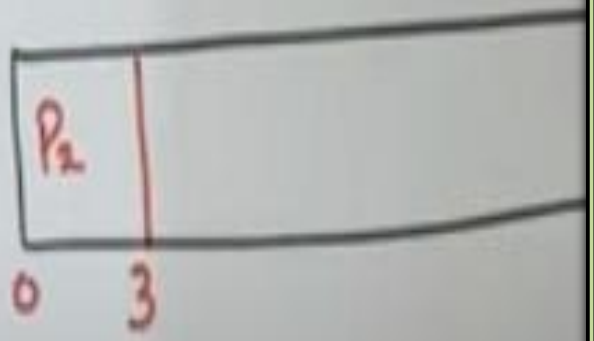
ble. Calculate Average Waiting time for the

owing:-

Process	Burst Time	Priority
P_1	21	2
P_2	3	1
P_3	6	4
P_4	2	3

Assume all Process arrives at 0.

Gantt chart



1) Problem of Starvation may occur.

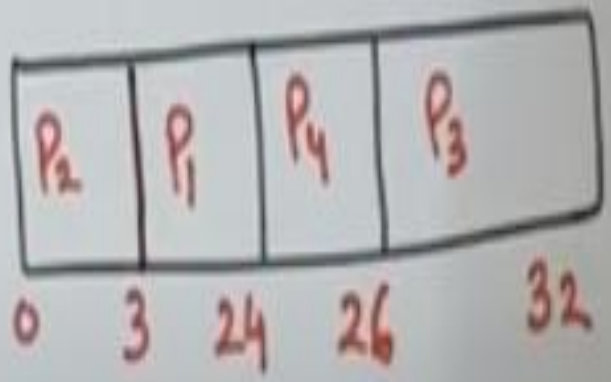
e. Calculate Average Waiting time for the

wing:-

Process	Burst Time	Priority
P_1	21	2
P_2	3	1
P_3	6	4
P_4	2	3

Assume all Process arrives at 0.

Gantt chart



iii) Problem of Starvation may occur.

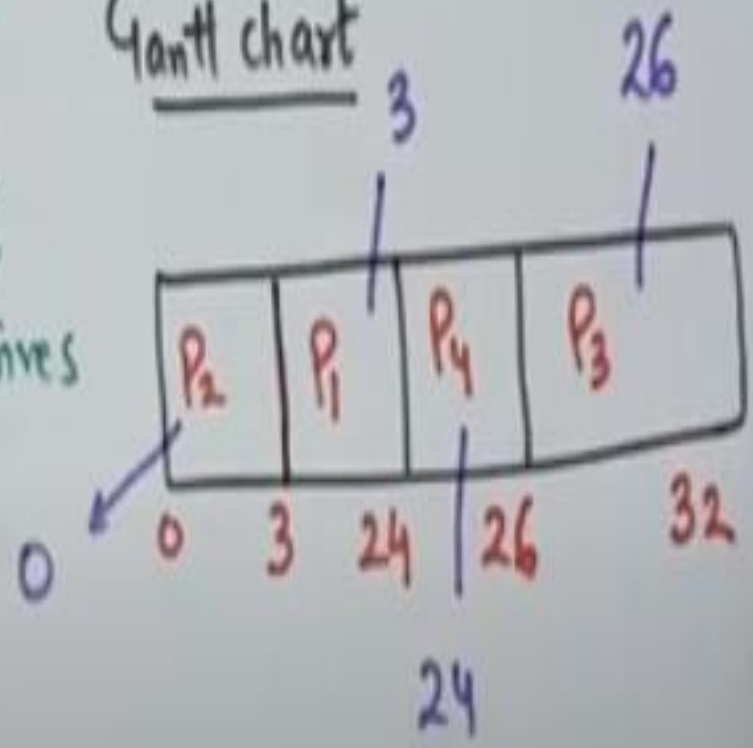
ple. Calculate Average Waiting time for the

wing:

Process	Burst Time	Priority
P_1	21	2
P_2	3	1
P_3	6	4
P_4	2	3

Assume all Process arrives at 0.

Gantt chart



FCFS manner.

P_4 → Aging (increases priority of process)

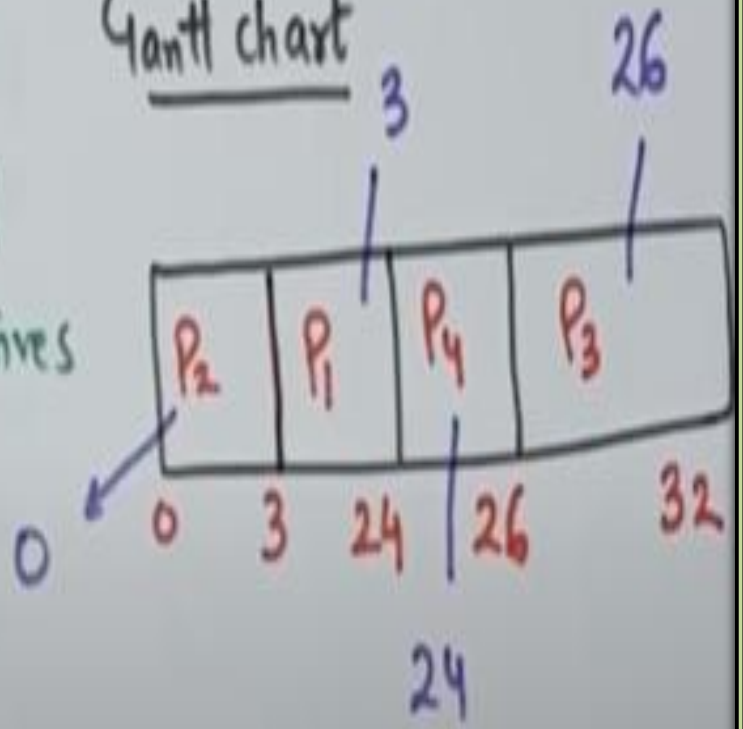
iiii) Problem of Starvation may occur.

Example. Calculate Average Waiting time for the following:

Process	Burst Time	Priority
P_1	21	2
P_2	3	1
P_3	6	4
P_4	2	3

Assume all Process arrives at 0.

Gantt chart



Avg wt: $\frac{0+3+24+26}{4} = 13.25$ ms.

Example of Priority Scheduling Algorithm

Consider the below table fo processes with their respective CPU burst times and the priorities.

PROCESS	BURST TIME	PRIORITY
P1	21	2
P2	3	1
P3	6	4
P4	2	3

The GANTT chart for following processes based on Priority scheduling will be,

The average waiting time will be, (0 + 3 + 24 + 26)/4 = 13.25 ms

As you can see in the GANTT chart that the processes are given CPU time just on the basis of the priorities.

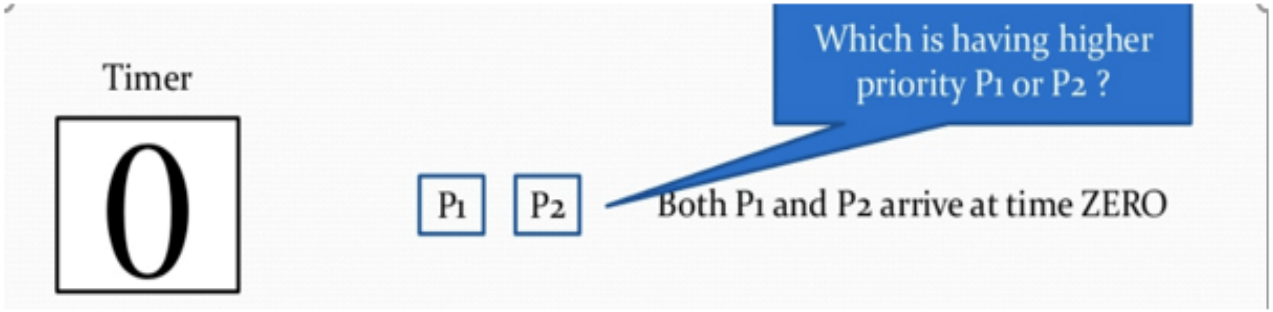
Example of Priority Scheduling(Preemptive)

Example of Priority Scheduling

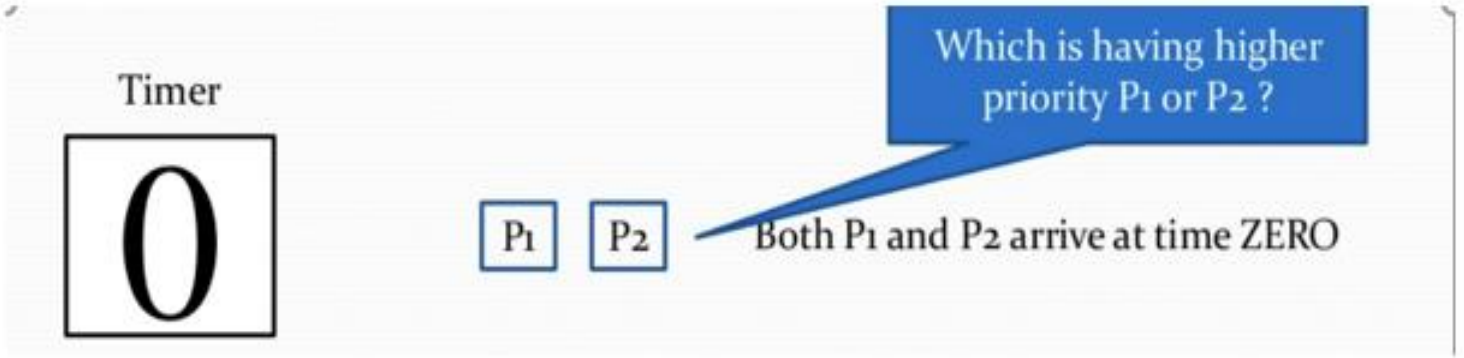
Consider following five processes P1 to P5. Each process has its unique priority, burst time, and arrival time.

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

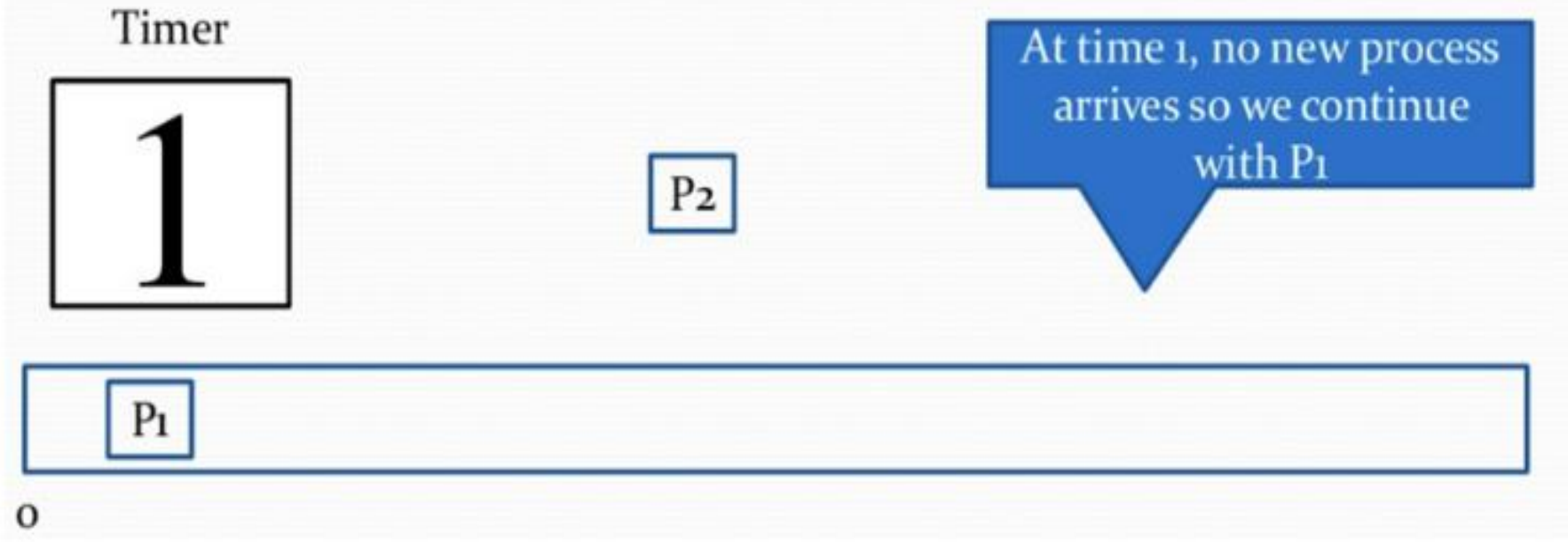
Step 0) At time=0, Process P1 and P2 arrive. P1 has higher priority than P2. The execution begins with process P1, which has burst time 4.



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.



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

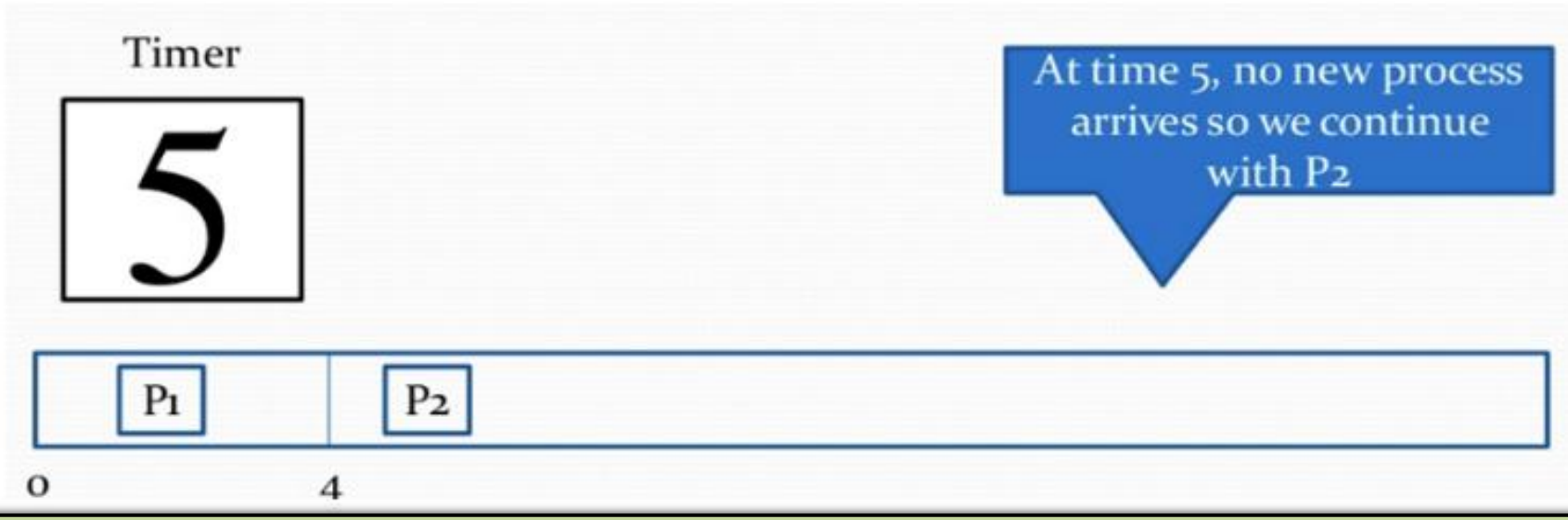


Step 2) At time 2, no new process arrives, so you can continue with P1. P2 is in the waiting queue.

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

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

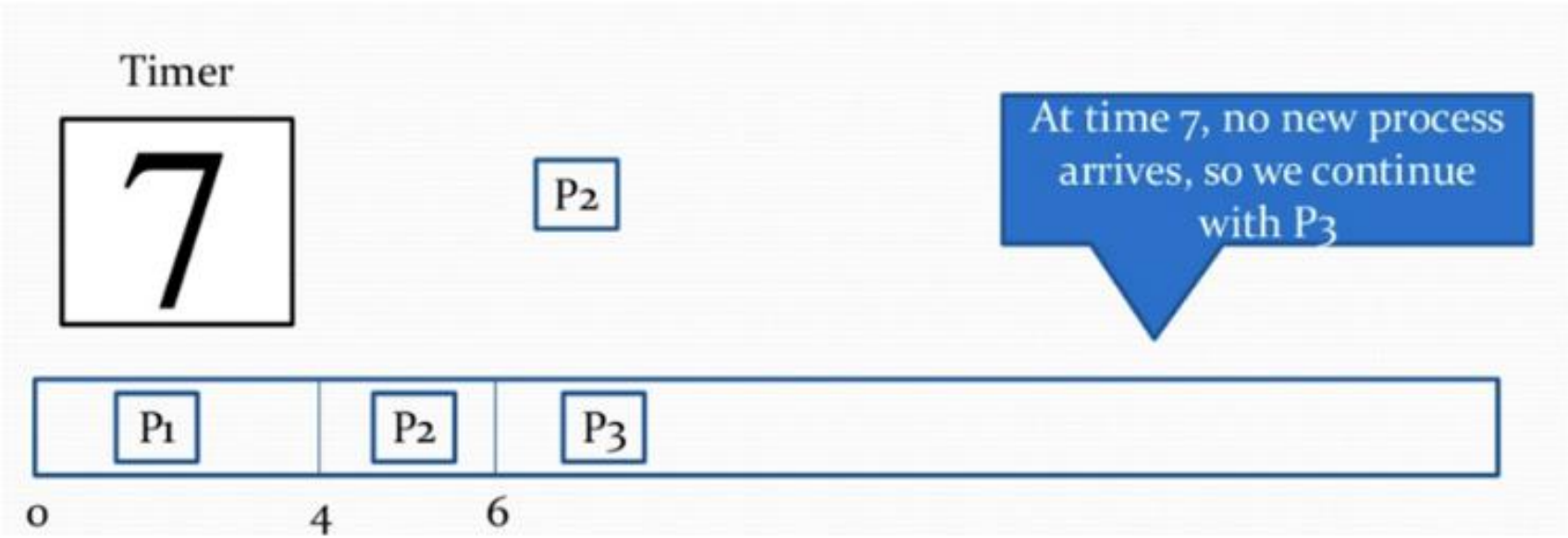
Step 5) At time= 5, no new process arrives, so we continue with P2.



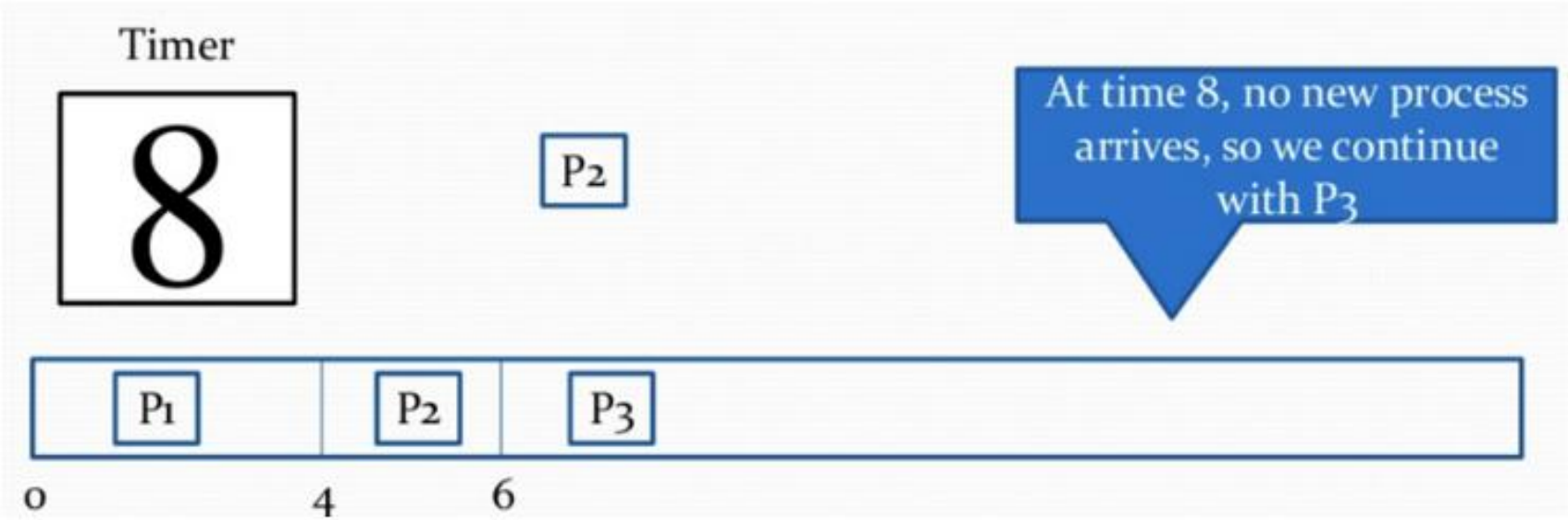
Step 6) At time=6, P3 arrives. P3 is at higher priority (1) compared to P2 having priority (2). P2 is preempted, and P3 begins its execution.

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

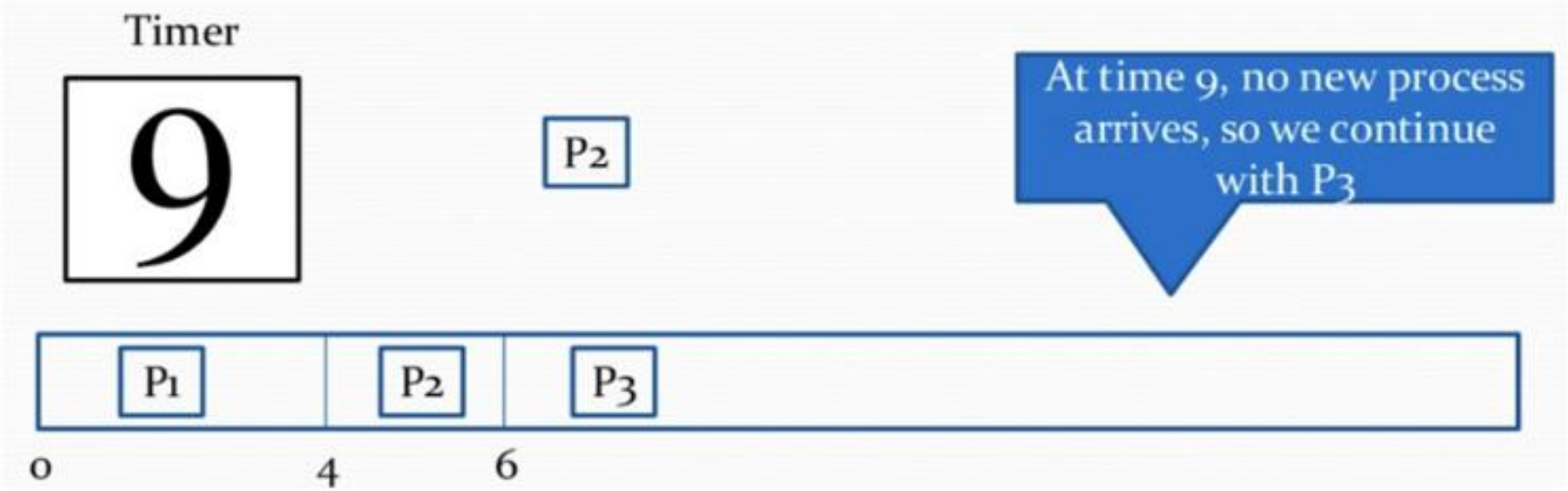
Step 7) At time 7, no-new process arrives, so we continue with P3. P2 is in the waiting queue.



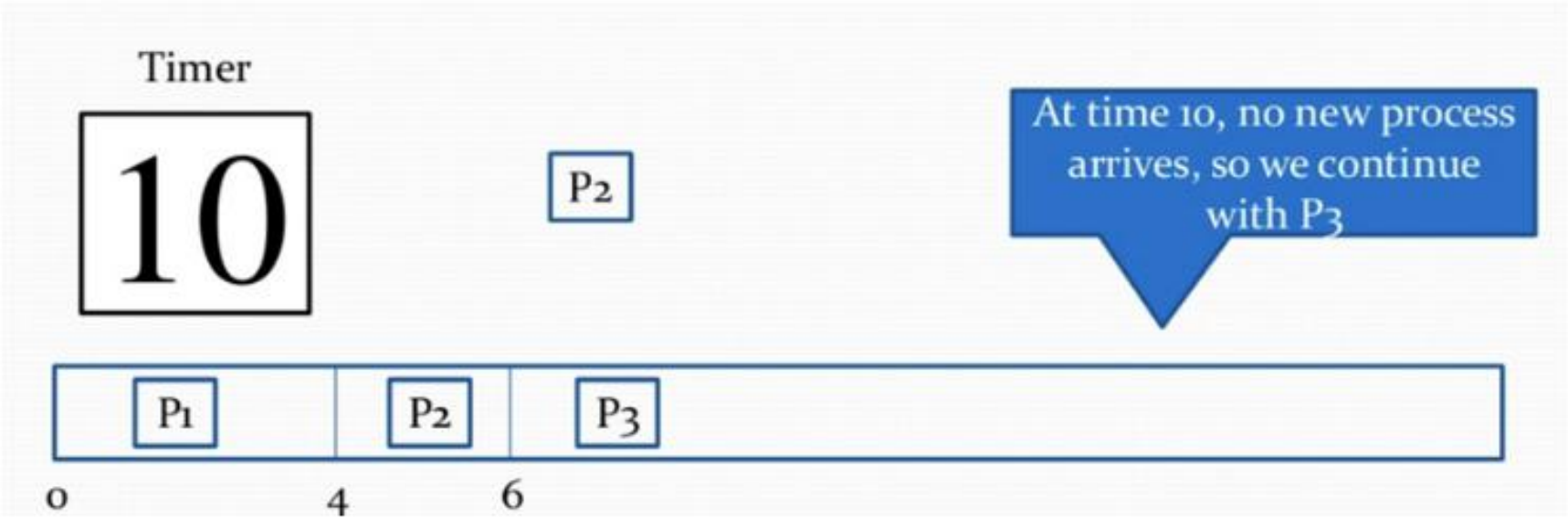
Step 8) At time= 8, no new process arrives, so we can continue with P3.



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

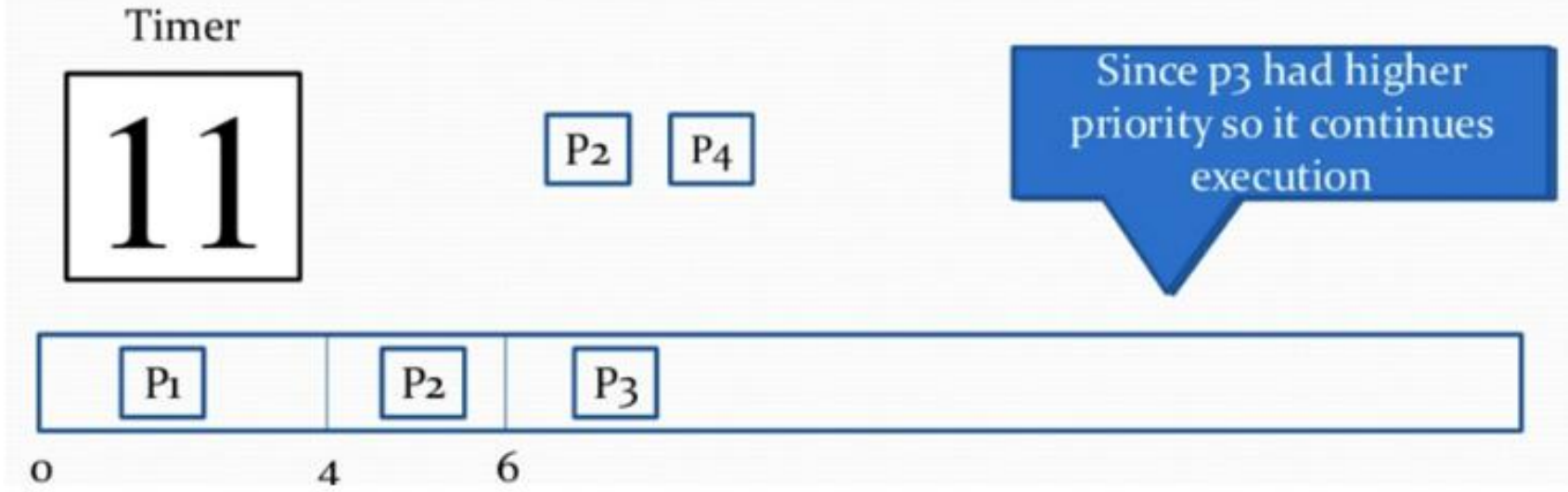


Step 10) At time interval 10, no new process comes, so we continue with P3



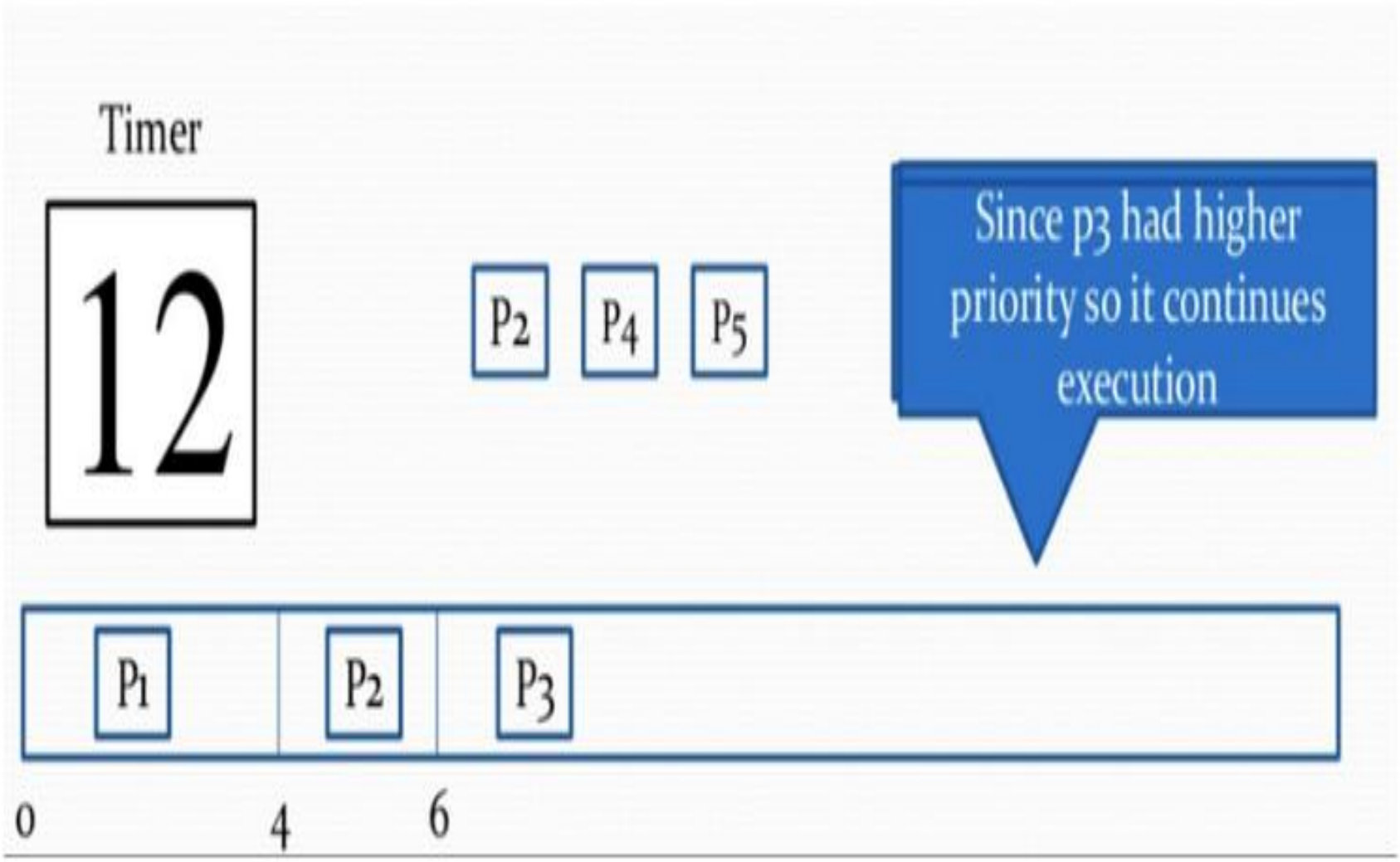
Step 11) At time=11, P4 arrives with priority 4. P3 has higher priority, so it continues its execution.

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



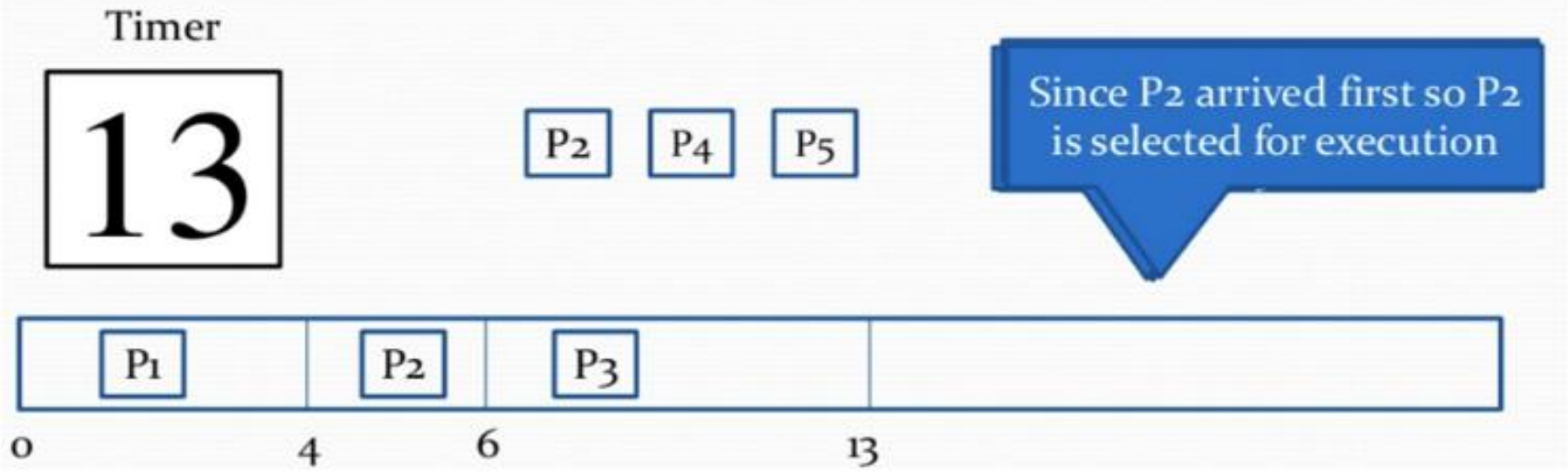
Step 12) At time=12, P5 arrives. P3 has higher priority, so it continues execution.

Step 12) At time=12, P5 arrives. P3 has higher priority, so it continues execution.

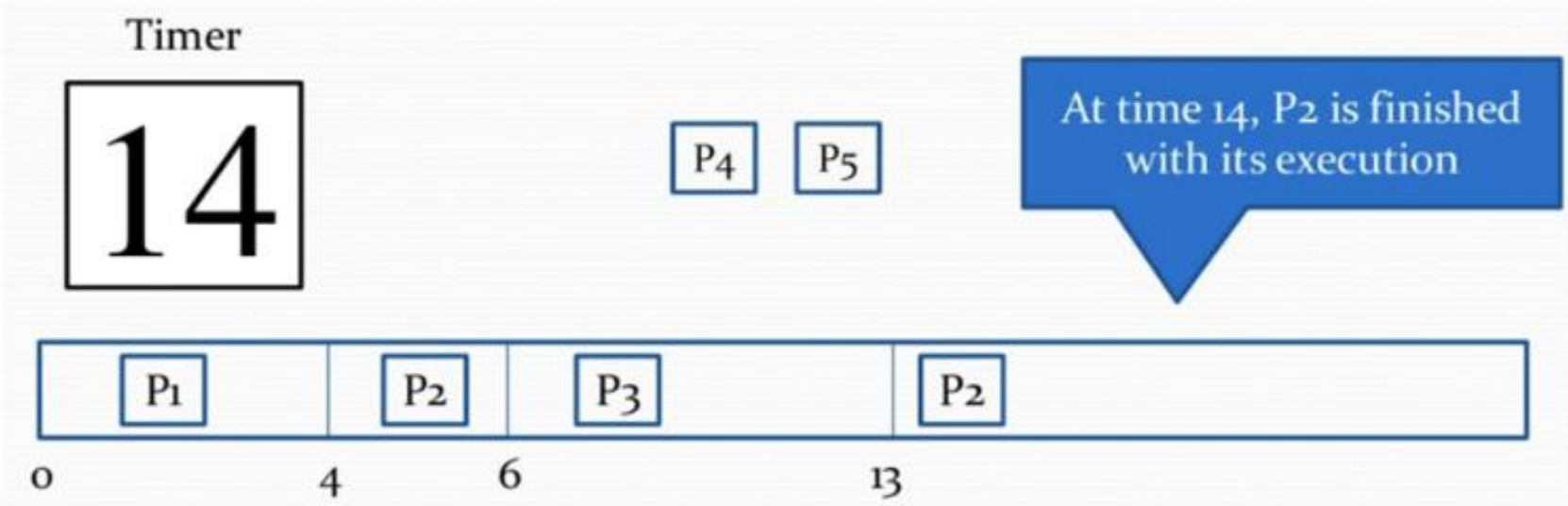


Step 13) At time=13, P3 completes execution. We have P2,P4,P5 in ready queue. P2 and P5 have equal priority. Arrival time of P2 is before P5. So P2 starts execution.

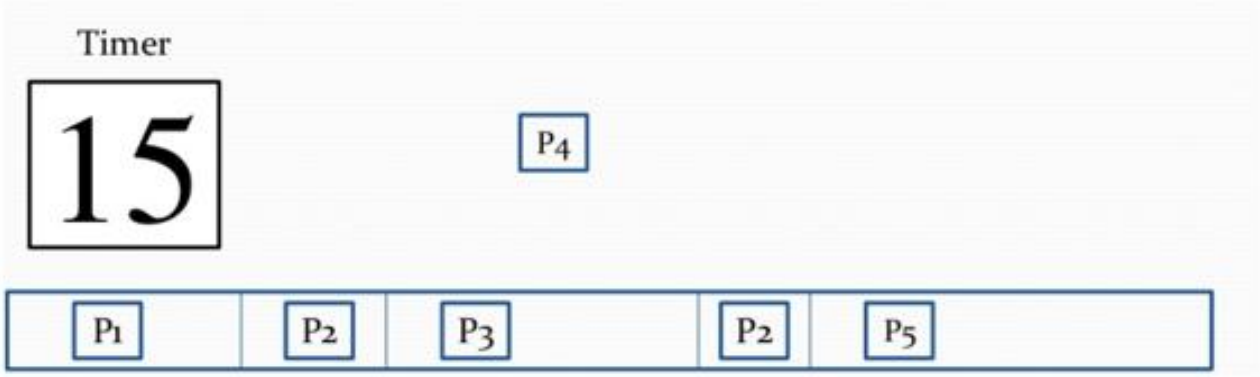
Process	Priority	Burst time	Arrival time
P1	1	4	0
P2	2	1 out of 3 pending	0
P3	1	7	6
P4	3	4	11
P5	2	2	12



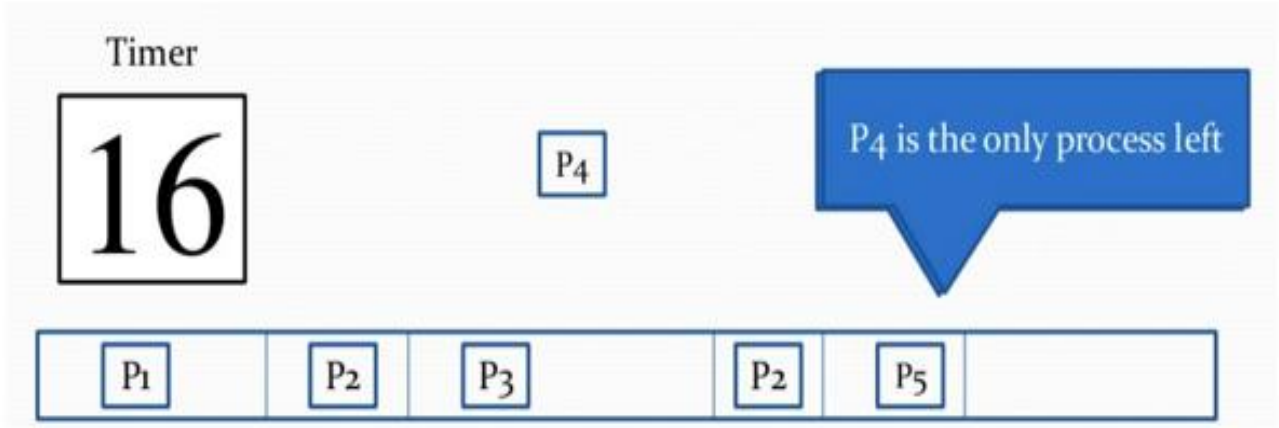
Step 14) At time =14, the P2 process has finished its execution. P4 and P5 are in the waiting state. P5 has the highest priority and starts execution.



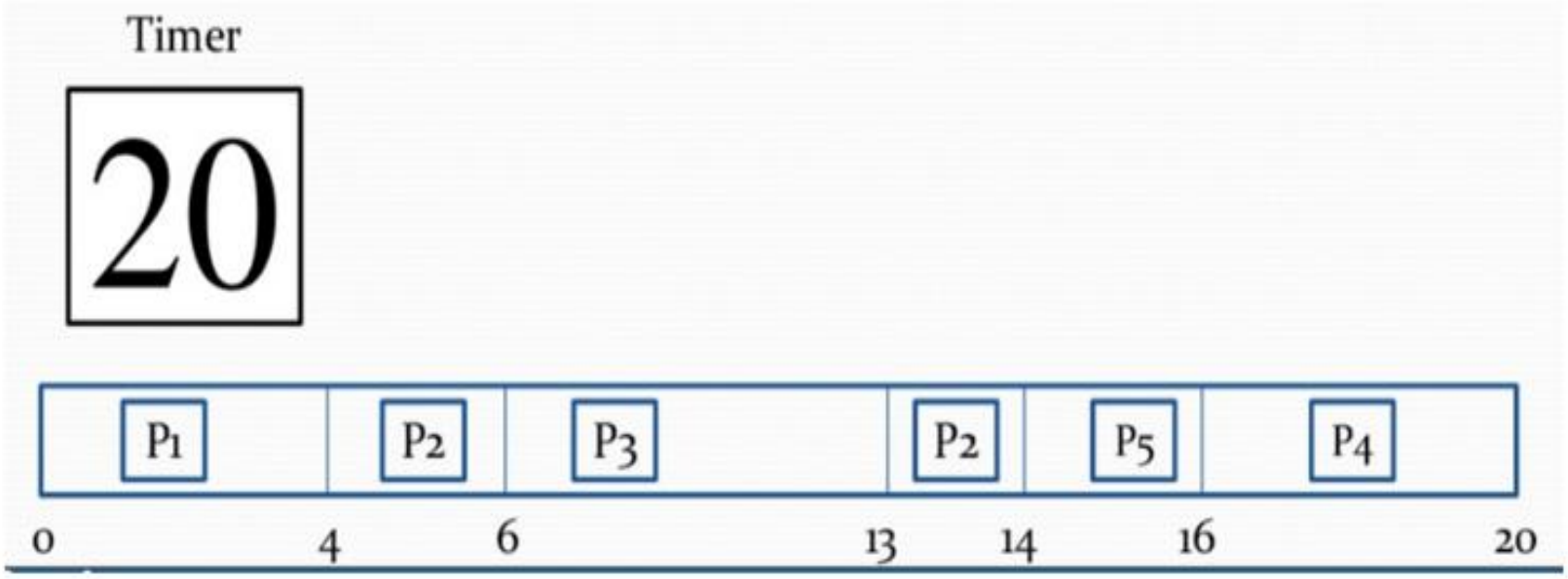
Step 15) At time =15, P5 continues execution.



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



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



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.

Summary:

- 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.
- In Priority Preemptive Scheduling, the tasks are mostly assigned with their priorities.
- In Priority Non-preemptive scheduling method, the CPU has been allocated to a specific process.
- Processes are executed on the basis of priority so high priority does not need to wait for long which saves time
- If high priority processes take lots of CPU time, then the lower priority processes may starve and will be postponed for an indefinite time.

Non-Preemptive Priority Scheduling(Practice Questions)

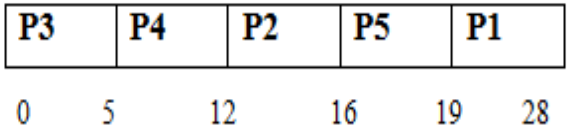
Non-Preemptive:

Consider the following processes and their CPU burst time and find out average waiting time and average turnaround time using priority scheduling algorithm (Lower number represents higher priority).

Process	Burst Time (mills.)	Priority
P1	9	5
P2	4	3
P3	5	1
P4	7	2
P5	3	4
Total	**28**	

- Since all processes comes simultaneously our selection will be easy based on non-preemptive scheduling scheme.
- According to priority scheme, process having highest priority will get the CPU first. In our example, process P3 has the highest priority (lower number) so P3 will be selected first for execution so waiting time for P3 will be zero. It will execute for five milliseconds after that next highest priority process is P4 so it will get the CPU next and waiting time for P4 will be five milliseconds i.e. the completion time of P3.
- Next turn is for process P2 after the completion of both process P3 and P4 so waiting time for process P2 will be 5 + 7(burst time of P3 and P4) = 12 milliseconds. Next comes process P5 with priority number 4 and having burst time of 3 milliseconds. Waiting time for P5 will be 5 + 7 + 4 = 16.

- Finally process P1 will get the CPU after waiting for 19 milliseconds (5 + 7 + 4 + 3). It will execute for 9 milliseconds so all process will be completed in total 28 milliseconds (i.e. 5 + 7 + 4 + 3 + 9 = 28).
- It can be represented in *gantt* chart as follow:



T.W.T= 19 + 12 + 0 + 5 + 16 =**52 mills**
A.W.T = 52 / 5 = **10.4 mills**
T.T.T = 28 + 16 + 5 + 12 + 19 = **80 mills.**
A.T.T =80 / 5 = **16 mills.**

Notation:

T.W.T = *Total Waiting Time*

A.W.T = *Average Waiting Time*

T.T.T = *Total Turnaround Time*

A.T.T = *Average Turnaround Time*

In the Non Preemptive Priority scheduling, The Processes are scheduled according to the priority number assigned to them. Once the process gets scheduled, it will run till the completion. Generally, the lower the priority number, the higher is the priority of the process. The people might get confused with the priority numbers, hence in the GATE, there clearly mention which one is the highest priority and which one is the lowest one.

Example

In the Example, there are 7 processes P1, P2, P3, P4, P5, P6 and P7. Their priorities, Arrival Time and burst time are given in the table.

Process ID	Priority	Arrival Time	Burst Time
1	2	0	3
2	6	2	5
3	3	1	4
4	5	4	2
5	7	6	9
6	4	5	4
7	10	7	10

We can prepare the Gantt chart according to the Non Preemptive priority scheduling.

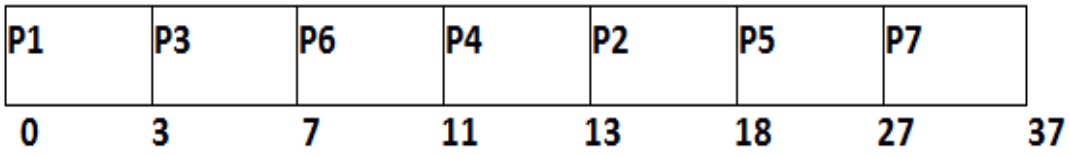
The Process P1 arrives at time 0 with the burst time of 3 units and the priority number 2. Since No other process has arrived till now hence the OS will schedule it immediately.

Meanwhile the execution of P1, two more Processes P2 and P3 are arrived. Since the priority of P3 is 3 hence the CPU will execute P3 over P2.

Meanwhile the execution of P3, All the processes get available in the ready queue. The Process with the lowest priority number will be given the priority. Since P6 has priority number assigned as 4 hence it will be executed just after P3.

After P6, P4 has the least priority number among the available processes; it will get executed for the whole burst time.

Since all the jobs are available in the ready queue hence All the Jobs will get executed according to their priorities. If two jobs have similar priority number assigned to them, the one with the least arrival time will be executed.



From the GANTT Chart prepared, we can determine the completion time of every process. The turnaround time, waiting time and response time will be determined.

Turn Around Time = Completion Time - Arrival Time

Waiting Time = Turn Around Time - Burst Time

Process Id	Priority	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time	Response Time
1	2	0	3	3	3	0	0
2	6	2	5	18	16	11	13
3	3	1	4	7	6	2	3
4	5	4	2	13	9	7	11
5	7	6	9	27	21	12	18
6	4	5	4	11	6	2	7
7	10	7	10	37	30	18	27

Avg Waiting Time = (0+11+2+7+12+2+18)/7 = 52/7 units

Assignment#04

Questions on Priority Based CPU Scheduling(Non-Preemptive)

Ques1. Calculate Waiting Time and Turn Around Time

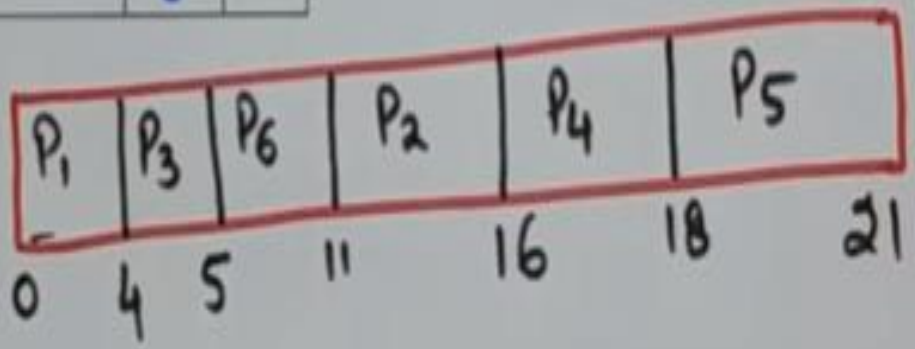
Process Number	Arrival Time	Burst Time	Priority	Completion Time	T.A.T	W.T
P1	0	4	4			
P2	1	5	5			
P3	2	1	7			
P4	3	2	2			
P5	4	3	1			
P6	5	6	6			

Questions on Priority Based CPU Scheduling(Non-Preemptive)

Ques1. Calculate Waiting Time and Turn Around Time

Process Number	Arrival Time	Burst Time	Priority	Completion Time	T.A.T	W.T
P1	0	4	4	4	4	
P2 ✔	1	5	5	16	15	
P3 ✔	2	1	7(H	5	3	
P4 ✔	3	2	2	18	15	
P5 ✔	4	3	1(L	21	17	
P6 ✔	5	6	6	11	6	

Gantt chart :-



Questions on Priority Based CPU Scheduling(Preemptive)

Ques1. Calculate Waiting Time and Turn Around Time

Process Number	Arrival Time	Burst Time	Priority	Completion Time	T.A.T	W.T
P1	1	4	4			
P2	2	2	5			
P3	2	3	7			
P4	3	5	8			
P5	3	1	5			
P6	4	2	6			

Questions on Priority Based CPU Scheduling(Preemptive)

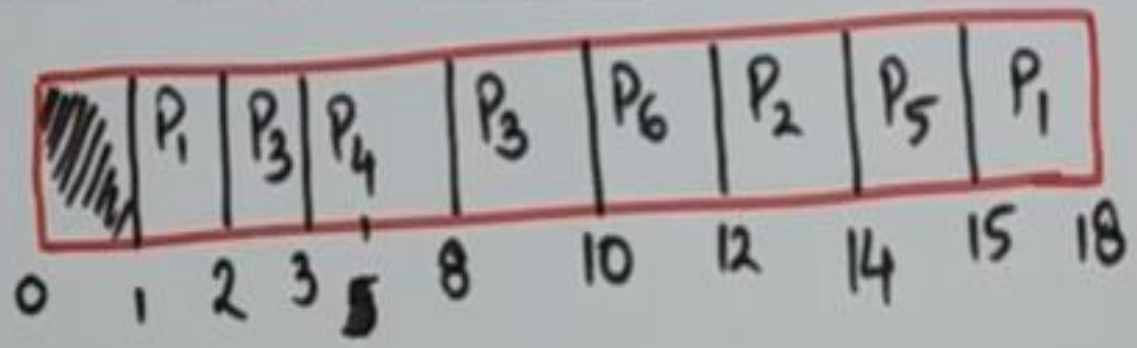
Ques1. Calculate Waiting Time and Turn Around Time

Process Number	Arrival Time	Burst Time	Priority	Completion Time	T.A.T	W.T
P1	1	4	4	18		
P2	2	2	5	14		
P3	2	3	7	10		
P4	3	5	8	8		
P5	3	1	5	15		
P6	4	2	6	12		

$P_1(3)$

$P_3(2)$

Gantt Chart



Priority Scheduling

Priority based Scheduling Algorithm

In Priority based Scheduling, each process is assigned a priority. Process with highest priority is to be executed first and so on. Processes with same priority are executed on first come first serve basis. Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Question

Implement Priority CPU Scheduling Algorithm using C,C++. Also attach the snapshot of Code and output window.

Implementation and Algorithm

AIM: To write a C program to implement the CPU scheduling algorithm for Priority.

PROBLEM DESCRIPTION:

Cpu scheduler will decide which process should be given the CPU for its execution. For this it uses different algorithm to choose among the process. One among that algorithm is Priority algorithm.

In this algorithm the processes will be given the priorities. The process which is having the highest priority is allocated the cpu first.

After finishing the request the cpu is allocated to the next highest priority and so on.

Algorithm

ALGORITHM:

Step 1: Get the number of process
Step 2: Get the id and service time for each process.
Step 3: Initially the waiting time of first short process as 0 and total time of first short is process the service time of that process.
Step 4: Calculate the total time and waiting time of remaining process.
Step 5: Waiting time of one process is the total time of the previous process.
Step 6: Total time of process is calculated by adding the waiting time and service time of each process.
Step 7: Total waiting time calculated by adding the waiting time of each process.
Step 8: Total turn around time calculated by adding all total time of each process.
Step 9: Calculate average waiting time by dividing the total waiting time by total number of process.
Step 10: Calculate average turn around time by dividing the total waiting time by total number of process.
Step 11: Display the result.

Implementation

PROGRAM:

```
void main()
{
        int i,j,n,t,turn[20],burst[20],p[20],wt[20],c[20];
        float await,aturn,twait=0,tturn=0;
        printf("\nEnter the value of n:");
        scanf("%d",&n);
        printf("\n Enter the process no burst and arrivaltime");
        for(i=0;i<n;i++)
        {
                scanf("%d",&c[i]);
                scanf("%d",&burst[i]);
```

```
                scanf("%d",&p[i]);
        }
        for(i=0;i<n;i++)
                for(j=i+1;j<n;j++)
                {
                        if(p[i]>p[j])
                        {
                                t=p[i];
                                p[i]=p[j];
                                p[j]=t;
                                t=burst[i];
                                burst[i]=burst[j];
                                burst[j]=t;
                                t=c[i];
                                c[i]=c[j];
                                c[j]=t;

                        }
                }
```

```
for(i=0;i<n;i++)
{
        if(i==0)
        {
                wt[i]=0;
                turn[i]=burst[i];

        }
}
else
{
        turn[i]=turn[i-1]+burst[i];
        wt[i]=turn[i]-burst[i];
        twait=twait+wt[i];
        tturn=tturn+turn[i];

}
```

```
await=twait/n;
aturn=tturn/n;
printf("pno\tbtime\tatime\twtime\tttime");
for(i=0;i<n;i++)
{
        printf("\n%d\t%d\t%d\t%d\t%d\n",c[i],burst[i],p[i],wt[i],turn[i]);
}
printf("\n The average waiting time is:%f",await);
printf("\n The average turn around time is:%f",aturn);
}
```

Output

Enter the no. of processes : 3 Enter the
burst times
P1 : 24
P2 : 5
P3 : 3
Enter the priorities

P1 : 2

P2 : 1

P3 : 3

Gantt Chart : P2 P1P3

ProcessID	Priority	BurstTime	WaitingTime	TurnaroundTime
P2	1	5	0	5
P1	2	24	5	29
P3	3	3	29	32

Average Waiting Time : 11.33

Average Turnaround Time : 22.00

Project | Classes | Debug

PriorityCPU.cpp

```cpp
#include<iostream>

using namespace std;

int main()
{
    int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg_wt,avg_tat;
    cout<<"Enter Total Number of Process:";
    cin>>n;

    cout<<"\nEnter Burst Time and Priority\n";
    for(i=0;i<n;i++)
    {
        cout<<"\nP["<<i+1<<"]\n";
        cout<<"Burst Time:";
        cin>>bt[i];
        cout<<"Priority:";
        cin>>pr[i];
        p[i]=i+1;           //contains process number
    }
```

```
21
22      //sorting burst time, priority and process number in ascending order using selection sort
23      for(i=0;i<n;i++)
24      {
25          pos=i;
26          for(j=i+1;j<n;j++)
27          {
28              if(pr[j]<pr[pos])
29                  pos=j;
30          }
31
32          temp=pr[i];
33          pr[i]=pr[pos];
34          pr[pos]=temp;
35
36          temp=bt[i];
37          bt[i]=bt[pos];
38          bt[pos]=temp;
39
40          temp=p[i];
41          p[i]=p[pos];
42          p[pos]=temp;
43      }
44
45      wt[0]=0;            //waiting time for first process is zero
46
```

```
    //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;

    cout<<"\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];
        cout<<"\nP["<<p[i]<<"]\t\t  "<<bt[i]<<"\t\t    "<<wt[i]<<"\t\t\t"<<tat[i];
    }

    avg_tat=total/n;     //average turnaround time
    cout<<"\n\nAverage Waiting Time="<<avg_wt;
    cout<<"\nAverage Turnaround Time="<<avg_tat;

    return 0;
}
```

```
Enter Total Number of Process:4

Enter Burst Time and Priority

P[1]
Burst Time:21
Priority:2

P[2]
Burst Time:3
Priority:1

P[3]
Burst Time:6
Priority:4

P[4]
Burst Time:2
Priority:3

Process      Burst Time            Waiting Time     Turnaround Time
P[2]                3                     0                  3
P[1]                21                    3                  24
P[4]                2                     24                 26
P[3]                6                     26                 32

Average Waiting Time=13
Average Turnaround Time=21
--------------------------------
Process exited after 40.8 seconds with return value 0
Press any key to continue . . .
```

Task

Implement Pre-emptive Priority CPU Scheduling Algorithm using C,C++. Also attach the snapshot of Code and output window.

Attach Task Output