



Operating System CSE316

Priority Scheduling Algorithm with context Switching time

Student Name: Subhrajit Nath

Student ID: 11810443

Section: K18FG

Email ID: gpro005@gmail.com

GitHub Link: <https://github.com/s33zar/OS-CA2>

QUESTION NO 14:

Write a program to implement priority scheduling algorithm with context switching time. Prompt to user to enter the number of processes and then enter their priority, burst time and arrival time also. Now whenever operating system pre-empt a process and shifts CPU's control to some another process of higher priority assume that it takes 2 seconds for context switching(dispatcher latency).Form a scenario, where we can give the processes are assigned with priority where the lower integer number is higher priority and then context switch .. as the process waits the priority of the process increase at rate of one per 2 time units of wait. Calculate waiting time and turnaround time for each process.

Description of problem:

- Priority scheduling is a non-pre-emptive algorithm and one of the most common scheduling algorithms in batch systems. Each process is assigned a priority. Process with the highest priority is to be executed first and so on. 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.

ALGORITHM:

1. Get the user input with choice
2. In this, each process is assigned a priority and processes are executed on the basis of their priority.
3. We can either choose to set priority of the lowest number to be the first priority.
4. We can either choose to set priority of the highest number to be the first priority.
5. No other process can execute until the process with the highest priority has fully executed.
6. If two processes have same priority, then process is executed on the basis of their arrival time.
7. Exit the program.

Program Snippet:

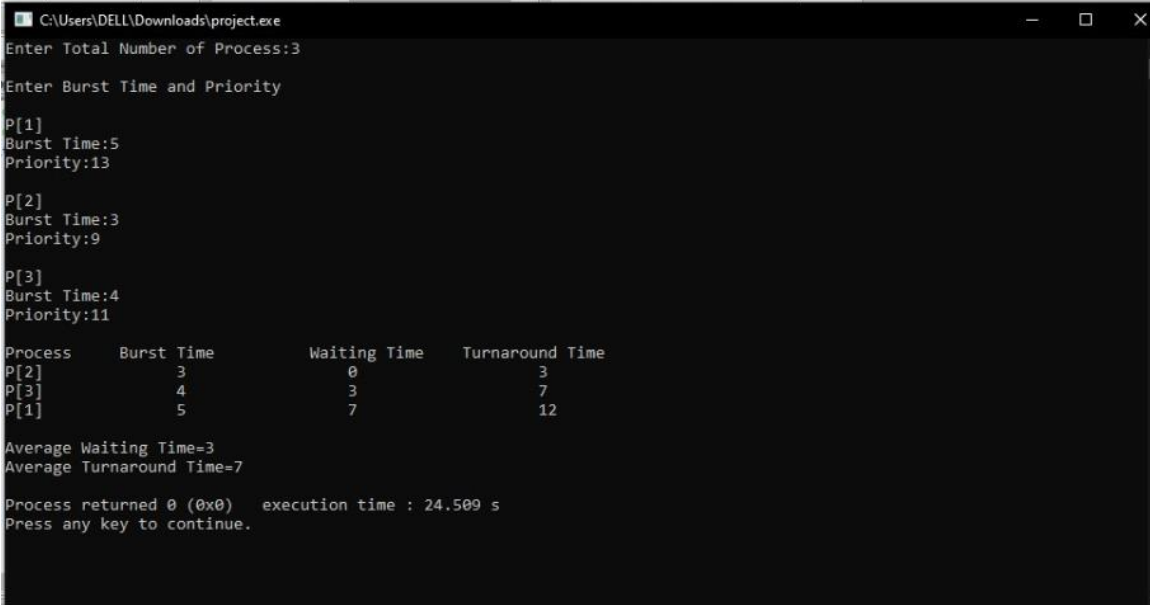
```
1  #include<stdio.h>
2  #include<conio.h>
3  int main()
4  {
5      int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg_wt,avg_tat;
6      printf("Enter Total Number of Process:");
7      scanf("%d",&n);
8
9      printf("\nEnter Burst Time and Priority\n");
10     for(i=0;i<n;i++)
11     {
12         printf("\nP[%d]\n",i+1);
13         printf("Burst Time:");
14         scanf("%d",&bt[i]);
15         printf("Priority:");
16
17         scanf("%d",&pr[i]);
18         p[i]=i+1;
19     }
20
21
22     for(i=0;i<n;i++)
23     {
24         pos=i;
25         for(j=i+1;j<n;j++)
26         {
27             if(pr[j]<pr[pos])
28                 pos=j;
29         }
30
31         temp=pr[i];
32         pr[i]=pr[pos];
33
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;
46
47     for(i=1;i<n;i++)
48     {
49         wt[i]=0;
50         for(j=0;j<i;j++)
51             wt[i]+=bt[j];
52
53         total+=wt[i];
54     }
55
56     avg_wt=total/n;
57     total=0;
58
59     printf("\nProcess\t\t Burst Time\t\t \tWaiting Time\tTurnaround Time");
60     for(i=0;i<n;i++)
61     {
62         tat[i]=bt[i]+wt[i];
63         total+=tat[i];
64         printf("\nP[%d]\t\t\t %d\t\t\t\t %d\t\t\t\t\t %d",p[i],bt[i],wt[i],tat[i]);
```

```

65
66     avg_tat=total/n;
67     printf("\n\nAverage Waiting Time=%d",avg_wt);
68     printf("\n\nAverage Turnaround Time=%d\n",avg_tat);
69
70     return 0;
71 }

```

Output



```

C:\Users\DELL\Downloads\project.exe
Enter Total Number of Process:3
Enter Burst Time and Priority
P[1]
Burst Time:5
Priority:13
P[2]
Burst Time:3
Priority:9
P[3]
Burst Time:4
Priority:11
Process    Burst Time    Waiting Time    Turnaround Time
P[2]       3             0              3
P[3]       4             3              7
P[1]       5             7             12
Average Waiting Time=3
Average Turnaround Time=7
Process returned 0 (0x0)   execution time : 24.509 s
Press any key to continue.

```

Complexity:

- Complexity of Priority Scheduling Algorithm: **O(n)**

Constraints:

In systems with power constraints, context switches in a task schedule result in wasted power consumption. We present variants of priority scheduling algorithms – Rate Monotonic and Earliest Deadline First – that reduce the number of context switches in a schedule. We prove that our variants output feasible schedules whenever the original algorithms do. We present experimental results to show that our variants significantly reduce the number of context switches. Our results also show that the number of context switches in the schedules output by these algorithms is close to the minimum possible number.

Boundary Conditions:

1. Process with highest priority gets preference first.
2. If two or more processes have same priority, then process is executed on their arrival time.
3. If there is already a process running and another process comes with higher priority, then the running process is pre-empted.

Test Cases:

1. Process 1 comes -> starts getting executed

```
Enter Total Number of Process:2

Enter Burst Time and Priority

P[1]
Burst Time:2
Priority:2

P[2]
Burst Time:3
Priority:3

Process      Burst Time      Waiting Time      Turnaround Time
P[1]          2                0                 2
P[2]          3                2                 5

Average Waiting Time=1
Average Turnaround Time=3
```

Conclusion: PASS

2. Next process comes with lower priority -> Waits till execution finishes

```
Enter Total Number of Process:2

Enter Burst Time and Priority

P[1]
Burst Time:3
Priority:1

P[2]
Burst Time:3
Priority:2

Process      Burst Time      Waiting Time      Turnaround Time
P[1]          3                0                  3
P[2]          3                3                  6

Average Waiting Time=1
Average Turnaround Time=4
```

Conclusion: PASS

3. Another process comes with higher priority -> Pre-empt's running process and starts executing

```
Enter Total Number of Process:2

Enter Burst Time and Priority

P[1]
Burst Time:2
Priority:1

P[2]
Burst Time:2
Priority:2

Process      Burst Time      Waiting Time      Turnaround Time
P[1]          2                0                  2
P[2]          2                2                  4

Average Waiting Time=1
Average Turnaround Time=3
```

Conclusion: PASS

4. If two process have similar priority -> process that arrives first, gets executed

```
Enter Total Number of Process:2
```

```
Enter Burst Time and Priority
```

```
P[1]
```

```
Burst Time:4
```

```
Priority:2
```

```
P[2]
```

```
Burst Time:4
```

```
Priority:2
```

Process	Burst Time	Waiting Time	Turnaround Time
P[1]	4	0	4
P[2]	4	4	8

```
Average Waiting Time=2
```

```
Average Turnaround Time=6
```

Conclusion: PASS