**LAB NO:4**

# Tribhuvan University

Institute of Engineering

## Pulchowk Campus


## Lab report on

Process Scheduling Algorithms

## Submitted by

Arpan Pokharel
075BCT015
Group: A

## Submitted to

Department of Electronics and Computer Engineering

**Submission Date:** Feb 13,2022

Theory

Process scheduling is an os task that schedules processes of different states like ready, waiting and running. Process scheduling allows os to allocate a time interval of CPU execution for each process. Another important reason for using a process scheduling system is that it keeps CPU busy all the time.

Scheduling Algorithms

1. First come first serve scheduling

As the name suggests, the process which arrives first, gets executed first or we can say that the process which request CPU first, gets the CPU allocated first. It is the non-preemptive type of scheduling. It is easy to understand and to implement.

2. Shortest Job First scheduling

In this algorithm, the job having shortest or less burst time will get the CPU first. It is the best approach to minimize the waiting time. It is the non-preemptive type of scheduling

3. Roond Robin scheduling

In this algorithm, the os defines a time quantum. All the process gets executed in the cyclic way. Each of the process will get the process for time quantum and get back to the ready queue to wait for it's next turn.

4. Shortest remaining time first

It is the preemptive form of SJF. In this algorithm, the os schedules the job according to the remaining time of execution.

## First Come First Serve

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

struct process {
    int pid;
    int bt;
    int wt, tt;
} p[10];

int main() {
    int i, n, totwt, tottt, avg1, avg2;
    printf("Enter the no. of process.\n");
    scanf("%d", &n);
    for (i = 1; i <= n; i++) {
        p[i].pid = i;
        printf("Enter the brust time");
        scanf("%d", &p[i].bt);
    }
    p[i].wt = 0;
    p[1].tt = p[1].bt + p[1].wt;
    i = 2;
    while (i <= n) {
        p[i].wt = p[i - 1].bt + p[i - 1].wt;
        p[i].tt = p[i].bt + p[i].wt;
        i++;
    }
    i = 1;
    totwt = tottt = 0;
    printf("\n processid \t bt \t wt \t tt \n");
    while (i <= n) {
        printf("\n\t%d \t %d \t %d \t %d", p[i].pid, p[i].bt, p[i].wt, p[i].tt);
        totwt = p[i].wt + totwt;
        tottt = p[i].tt + tottt;
        i++;
    }
    avg1 = totwt / n;
    avg2 = tottt / n;
    printf("\n avg = %d \t avg2 = %d \t", avg1, avg2);
    return 0;
}

```

## Output

```
PS C:\Users\Arpan\Desktop\Algorithm> .\FCFS.exe
Enter the no. of process.
5
Enter the brust time2
Enter the brust time5
Enter the brust time8
Enter the brust time4
Enter the brust time3

 processid       bt      wt      tt

        1        2       0       2
        2        5       2       7
        3        8       7       15
        4        4       15      19
        5        3       19      22
 avg = 8         avg2 = 13
```

## Shortest Job First

```c
#include<stdio.h>
 int main()
{
    int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
    float avg_wt,avg_tat;
    printf("Enter number of process:");
    scanf("%d",&n);

    printf("\nEnter Burst Time:\n");
    for(i=0;i<n;i++)
    {
        printf("p%d:",i+1);
        scanf("%d",&bt[i]);
        p[i]=i+1;
    }

   //sorting of burst times
    for(i=0;i<n;i++)
    {
        pos=i;
        for(j=i+1;j<n;j++)
        {
            if(bt[j]<bt[pos])
                pos=j;
        }

        temp=bt[i];
        bt[i]=bt[pos];
        bt[pos]=temp;

        temp=p[i];
        p[i]=p[pos];
        p[pos]=temp;
    }

    wt[0]=0;


    for(i=1;i<n;i++)
    {
        wt[i]=0;
        for(j=0;j<i;j++)
            wt[i]+=bt[j];

        total+=wt[i];
    }

    avg_wt=(float)total/n;
    total=0;

    printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];
        total+=tat[i];
        printf("\n%d\t\t  %d\t\t    %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
    }

    avg_tat=(float)total/n;
    printf("\n\nAverage Waiting Time=%f",avg_wt);
    printf("\nAverage Turnaround Time=%f\n",avg_tat);
}
```

## Output

```
PS C:\Users\Arpan\Desktop\Algorithm> .\shortestjob.exe
Enter number of process:5

Enter Burst Time:
p1:3
p2:7
p3:3
p4:2
p5:5

Process     Burst Time          Waiting Time    Turnaround Time
4                2                  0                  2
3                3                  2                  5
1                3                  5                  8
5                5                  8                  13
2                7                  13                 20

Average Waiting Time=5.600000
Average Turnaround Time=9.600000
```

## Shortest Remaining Time First

```
1.  #include<stdio.h>
2.  #include<stdlib.h>
3.  #include<conio.h>
4.
5.  int main()
6.  {
7.
8.    int count,i,j,m=0,n,y=0,time,remain=0,min,flag=0;
9.    int wait_time=0,turn_a_time=0,a_time[10],b_time[10],p[10],z[10];
10.   float k=0,x=0;
11.   printf("Enter number of Process:\t ");
12.   scanf("%d",&n);
13.   for(count=0;count<n;count++)
14.   {
15.     printf("Enter Arrival Time and Burst Time for Process Process Number %d :",count+1);
16.     scanf("%d",&a_time[count]);
17.     scanf("%d",&b_time[count]);
18.   }
19.
20.   printf("\n\n\tProcess\t|Turnaround Time|Waiting Time\n\n");
21.   printf("\t=========================================\n");
22. for(i=0;i<n;i++)
23. {
24.     m=m+b_time[i];
25. }
26. min=m;
27. time=m;
28. for(i=0;i<n;i++)
29. {
30.     if(a_time[i]<time)
31.     {
32.         time=a_time[i];
33.     }
34. }
35. for(i=time;i<=m;i=i+b_time[j])
36. {
37.     min=m;
38.     remain=0;
39.     flag=0;
40.
41.     for(count=0;count<n;count++)
42.     {
43.
44.         if(a_time[count]<=i)
45.         {
46.
47.             if(b_time[count]<min)
48.             {
49.
50.                 min = b_time[count];
51.                 j=count;
52.                 flag=1;
53.             }
54.             remain=1;
55.         }
56.     }
57.     if(flag==1&&remain==1)
58.     {
59.         wait_time=i-a_time[j];
60.         turn_a_time=wait_time+b_time[j];
61.         printf("\tP[%d]\t|\t%d\t|\t%d\n",j+1,turn_a_time,wait_time);
62.         k=k+wait_time;
63.         x=x+turn_a_time;
64.
65.         a_time[j]=m+1;
66.         p[y]=j+1;
```

```
67.         z[y]=i;
68.         y++;
69.     }
70. }
71. printf("\n\nAverage Waiting Time= %.2f\n",k/n);
72. printf("Avg Turnaround Time = %.2f",x/n);
73. printf("\n\n\nTotal time taken by processor to complete all the jobs : %d",m);
74. printf("\n\nQueue  for order of execution:\n");
75. printf("\n\nProcess     ");
76.
77. for(i=0;i<n;i++)
78. {
79.     printf(" P[%d]   ",p[i]);
80.     if(i==(n-1))
81.     {
82.         printf("End");
83.     }
84. }
85.
86.   return 0;
87. }
```

## Output

```
PS C:\Users\Arpan\Desktop\Algorithm> .\shortesttime.exe
Enter number of Process:        5
Enter Arrival Time and Burst Time for Process Process Number 1 :2
4
Enter Arrival Time and Burst Time for Process Process Number 2 :5
3
Enter Arrival Time and Burst Time for Process Process Number 3 :5
6
Enter Arrival Time and Burst Time for Process Process Number 4 :7
4
Enter Arrival Time and Burst Time for Process Process Number 5 :6
4


        Process |Turnaround Time|Waiting Time

        ========================================
        P[1]    |       4       |       0
        P[2]    |       4       |       1
        P[4]    |       6       |       2
        P[5]    |       11      |       7
        P[3]    |       18      |       12


Average Waiting Time= 4.40
Avg Turnaround Time = 8.60


Total time taken by processor to complete all the jobs : 21

Queue  for order of execution:


Process         P[1]    P[2]    P[4]    P[5]    P[3]   End
```

# Round Robin

```c
#include<stdio.h>

int main()
{

  int count,j,n,time,remain,flag=0,time_quantum;
  int wait_time=0,turnaround_time=0,at[10],bt[10],rt[10];
  printf("Enter Total Process:\t ");
  scanf("%d",&n);
  remain=n;
  for(count=0;count<n;count++)
  {
    printf("Enter Arrival Time and Burst Time for Process Process Number %d :",count+1);
    scanf("%d",&at[count]);
    scanf("%d",&bt[count]);
    rt[count]=bt[count];
  }
  printf("Enter Time Quantum:\t");
  scanf("%d",&time_quantum);
  printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
  for(time=0,count=0;remain!=0;)
  {
    if(rt[count]<=time_quantum && rt[count]>0)
    {
      time+=rt[count];
      rt[count]=0;
      flag=1;
    }
    else if(rt[count]>0)
    {
      rt[count]-=time_quantum;
      time+=time_quantum;
    }
    if(rt[count]==0 && flag==1)
    {
      remain--;
      printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-bt[count]);
      wait_time+=time-at[count]-bt[count];
      turnaround_time+=time-at[count];
      flag=0;
    }
    if(count==n-1)
      count=0;
    else if(at[count+1]<=time)
      count++;
    else
      count=0;
  }
  printf("\nAverage Waiting Time= %f\n",wait_time*1.0/n);
  printf("Avg Turnaround Time = %f",turnaround_time*1.0/n);

  return 0;
}

```

## Output

```
PS C:\Users\Arpan\Desktop\Algorithm> .\roundrobin.exe
Enter Total Process:     5
Enter Arrival Time and Burst Time for Process Process Number 1 :2
4
Enter Arrival Time and Burst Time for Process Process Number 2 :5
7
Enter Arrival Time and Burst Time for Process Process Number 3 :5
7
Enter Arrival Time and Burst Time for Process Process Number 4 :32
6
Enter Arrival Time and Burst Time for Process Process Number 5 :34
5
Enter Time Quantum:     2


Process |Turnaround Time|Waiting Time

P[1]    |       2       |       -2
```

## Discussion

- In program 1, we used the first come first serve process scheduling algorithm. In this algorithm, the process are scheduled on the basis of their arrival time. The waiting time and turn around time was calculated and their average was also displayed.

- In program 2, we used the shortest job first algorithm. In this algorithm the process are scheduled on the basis of burst time

- In program 3, we used the shortest remaining time first. In this algorithm the process are scheduled according to the remaining time of execution

- In program 4, we used the round robin algorithm for process scheduling. The time slice was defined 2 units.

## conclusion

Hence, in this lab we learned about various process scheduling algorithms.