Mawlana Bhashani Science and Technology University

Lab-Report

Lab Report No: 08

Lab Report Name: File Implementation of SJF Scheduling Algorithm.

Course code: ICT-3110

Course title: Operating System Lab

Date of Performance: 7/9/2020

Date of Submission:

Submitted by

Name: Afra Ibnat Tethye

ID: IT-18055

3rd year 1st semester

Session: 2017-18

Dept. of ICT

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT,

MBSTU.

Experiment Name: Implementation of SJF Scheduling Algorithm.

Question-01:

What is SJF Scheduling algorithm?

Answer:

SJF stands for Shortest job first. It is a scheduling algorithm in which the process with the smallest execution time is selected for execution next. Shortest job first can be either preemptive or nonpreemptive. Owing to its simple nature, shortest job first is considered optimal. It also reduces the average waiting time for other processes awaiting execution.

Question-02:

What is SJF Scheduling algorithm?

Answer:

SJF implemented code is given below:

```
#include<stdio.h>

using namespace std;



int  main()

{

   int n,b[40],p[30],w[30],ta[40],i,j,total=0,pos,tem;

float avgw,avgta;

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",&b[i]);        p[i]=i+1;
   }    for(i=0; i<n; i++)
   {        pos=i;
for(j=i+1; j<n; j++)
       {
          if(b[j]<b[pos])
pos=j;
       }
         tem=b[i];
b[i]=b[pos];
   b[pos]=tem;

       tem=p[i];
p[i]=p[pos];
p[pos]=tem;
```

```
   }


w[0]=0;
for(i=1; i<n; i++)
{       w[i]            =0;


for(j=0; j<i; j++)
w[i]+=b[j];


      total+=w[i];
   }
avgw=(float)total/n;
total=0;
printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");
for(i=0; i<n; i++)
   {       ta[i] =b[i]+w[i];
       total+=ta[i];
printf("\np%d\t\t %d\t\t    %d\t\t\t%d",p[i],b[i],w[i],ta[i]);
   }
```

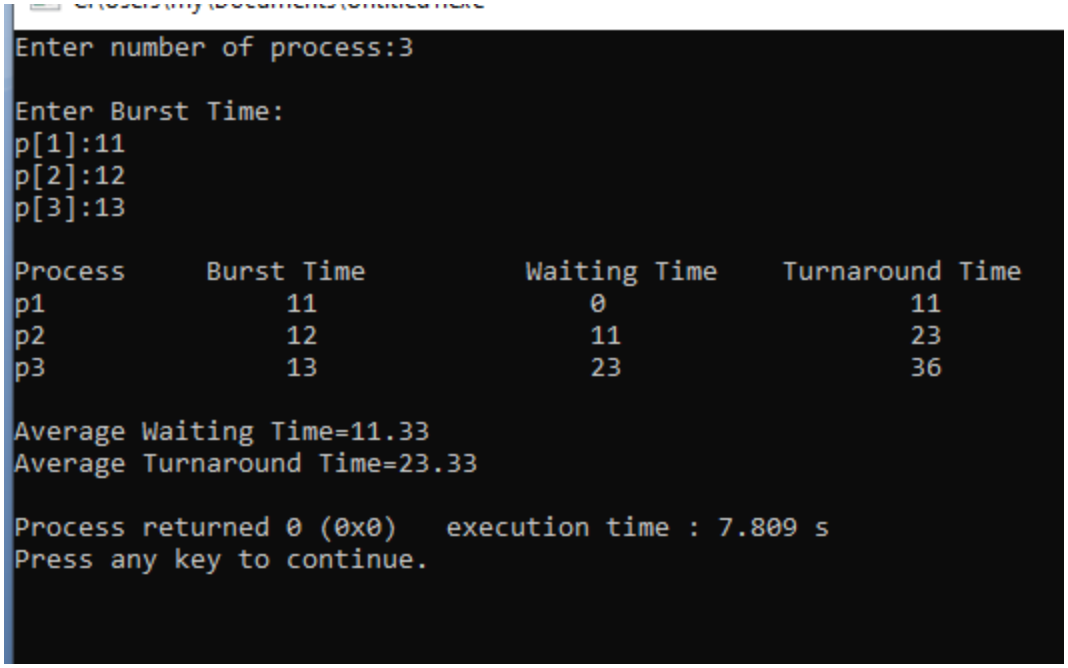
```
avgta=(float)total/n;

printf("\n\nAverage Waiting Time=%.2f",avgw);

printf("\nAverage Turnaround Time=%.2f\n",avgta);

}
```

Output:



Conclusion:

This is the best approach to minimize waiting time. To successfully implement it, the burst time/duration time of the processes should be known to the processor in advance, which is practically not feasible all the time. This scheduling algorithm is optimal if all the jobs/processes are available at the same time. (either Arrival time is 0 for all, or Arrival time is same for all)