Mawlana Bhashani Science and Technology University

Lab-Report

Report No: 08

Course code: ICT-3110

Course title: Operating Systems Lab

Date of Performance:

Date of Submission: 11/09/2020

Submitted by

Name: Golam Kibria Tuhin

ID:IT-18015

3th year 1ndsemester

Session: 2017-2018

Dept. of ICT

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Name of the lab report : Implementation of SJF Scheduling Algorithm.

Objective: SJF algorithm Definition & executable code in c are followed.

Q.1 What is SjF Scheduling algorithm?

Answer: Shortest job first 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.

Q.2 How to implemented in C?

Answer:

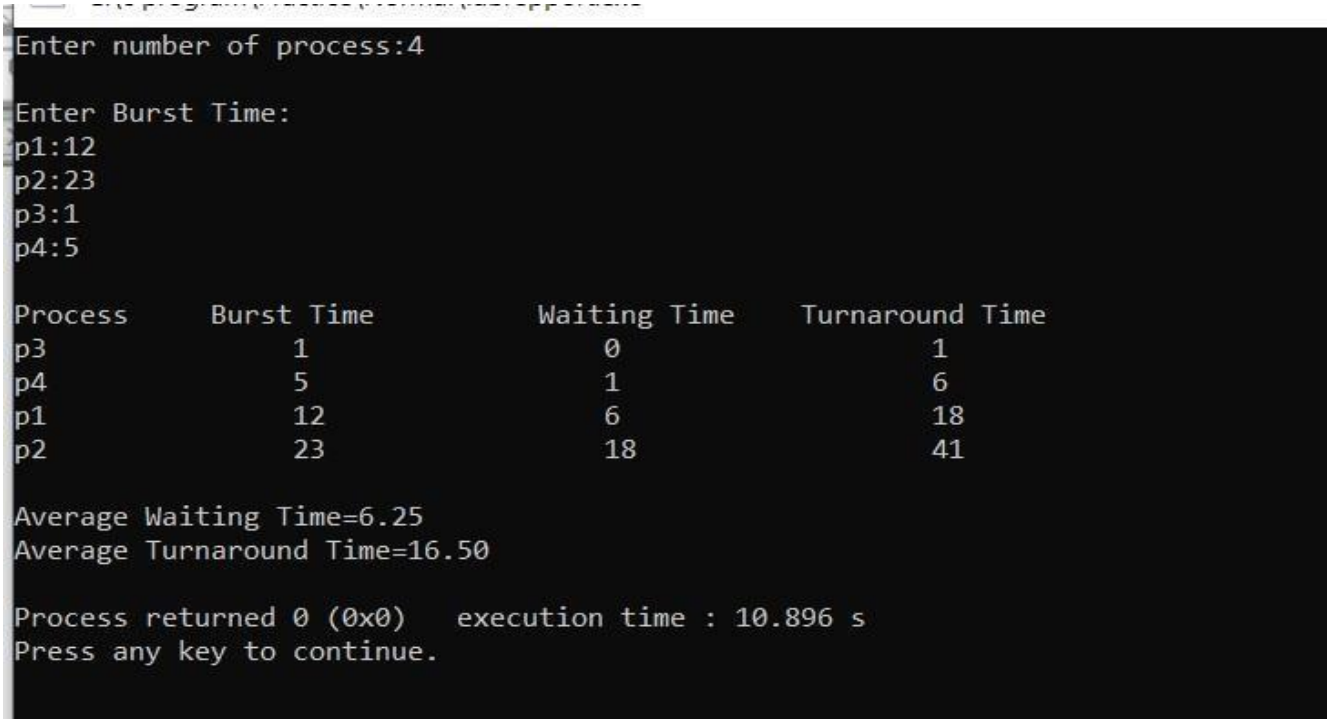
The code written in c are given below:

```
#include<stdio.h> using namespace

std;


int  main()

{

   int bt[40],p[30],wt[30],tat[40],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;


   }     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("\np%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=%.2f",avg_wt);
printf("\nAverage Turnaround Time=%.2f\n",avg_tat);
```

}

Output:



Conclusion:

Enhanced SJF eliminate the problem of starvation for bigger processes. Bigger processes will avail the chance to get the CPU on their turn even though the shorter processes are present in the Ready Queue. Main emphasis of this work is to ensure that both smaller processes and bigger processes get the CPU. Neither one process should monopolize the CPU as in FCFS nor should bigger processes starve. It has been ensured that the average waiting time of Enhanced SJF remains closer to the average waiting time of shortest job first and also that no job starve for CPU for longer time as does in shortest job first. Many bigger processes are urgent to execute but submitted at the end of the queue. No conventional algorithm give them chance to complete their execution in a certain time, In SJF they will not get the CPU until all shorter processes have been executed and in FCFS as they are at the back of the queue they will not get the CPU before their turn. But Enhanced SJF will allocate the CPU to that process on its turn while ensuring that it is done in some definite amount of time.