Mawlana Bhashani Science and Technology University

Lab-Report

Report No:08

Report Name:Implementation of SJF Scheduling algorithm

Course code:ICT-3110

Course title:Operating System Lab

Date of Performance:16-09-2020

Date of Submission:19-09-2020

Submitted by

Name:Sabikun Nahar Piya

ID:IT-18020

3rd year 1stsemester

Session: 2017-2018

Dept. of ICT

MBSTU.

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Experiment No:08

Experiment Name:Implementation of Sjf scheduling algorithm

Theory:

Shortest Job First (SJF) is an algorithm in which the process having the smallest execution time is chosen for the next execution. This scheduling method can be preemptive or non-preemptive. It significantly reduces the average waiting time for other processes awaiting execution. The full form of SJF is Shortest Job First.

Implementation:

Step 1: Declare the array size.

Step 2: Get the number of elements to be inserted.

Step 3: Select the process which have shortest burst will execute first.

Step 4: If two process have same burst length then FCFS scheduling algorithm used.

Step 5: Make the average waiting the length of next process.

Step 6: Start with the first process from it's selection as above and let other process to be in queue.

Step 7: Calculate the total number of burst time.

Step 8: Display the values.

Working process:

code for SJF scheduling algorithm:

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

Output:

```
Enter number of process:3

Enter Burst Time:
p1:4
p2:8
p3:3

Process     Burst Time          Waiting Time    Turnaround Time
p3                3                 0                 3
p1                4                 3                 7
p2                8                 7                 15

Average Waiting Time=3.333333
Average Turnaround Time=8.333333

Process returned 0 (0x0)   execution time : 35.159 s
Press any key to continue.
```

Discussion:

SJF is associated with each job as a unit of time to complete.This algorithm method is helpful for batch-type processing, where waiting for jobs to complete is not critical.It can improve process throughput by making sure that shorter jobs are executed first, hence possibly have a short turnaround time.It improves job output by offering shorter jobs, which should be executed first, which mostly have a shorter turnaround time.