Mawlana Bhashani Science and Technology University

Lab -Report

Report No:08
Course code: ICT-3110
Course title: Operating System Lab
Date of Performance:02-09-2020
Date of Submission:09-09-2020

Submitted by

Name: Ruku Shikder
ID: IT-18057
3th year 1st semester
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.

- What is SJF Scheduling Algorithm?
- How to implementation in C?

SJF Scheduling Algorithm:

Shortest job first(SJF) is a scheduling algorithm, that is used to schedule processes in an operating system. It is a very important topic in Scheduling when compared to round-robin and FCFS Scheduling. In this article, we will discuss the Shortest Job First Scheduling in the following order:

- Types of SJF
- Non-Preemptive SJF
- Code for Non-Preemptive SJF Scheduling
- Code for Pre-emptive SJF Scheduling

There are two types of SJF

- Pre-emptive SJF
- Non-Preemptive SJF

These algorithms schedule processes in the order in which the shortest job is done first. It has a minimum average waiting time.

There are 3 factors to consider while solving SJF, they are

- BURST Time
- Average waiting time
- Average turnaround time

Algorithm:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Start the Ready Q according the shortest Burst time by sorting according to lowest to highest burst time.

Step 5: Set the waiting time of the first process as‗0' and its turnaround time as its burst time.

Step 6: Sort the processes names based on their Burt time

Step 7: For each process in the ready queue, calculate

a) Waiting time(n)= waiting time (n-1) + Burst time (n-1)

b) Turnaround time (n)= waiting time(n)+Burst time(n) **Step 8:**

Calculate

a) Average waiting time = Total waiting Time / Number of process

b) Average Turnaround time = Total Turnaround Time / Number of process

Step 9: Stop the process

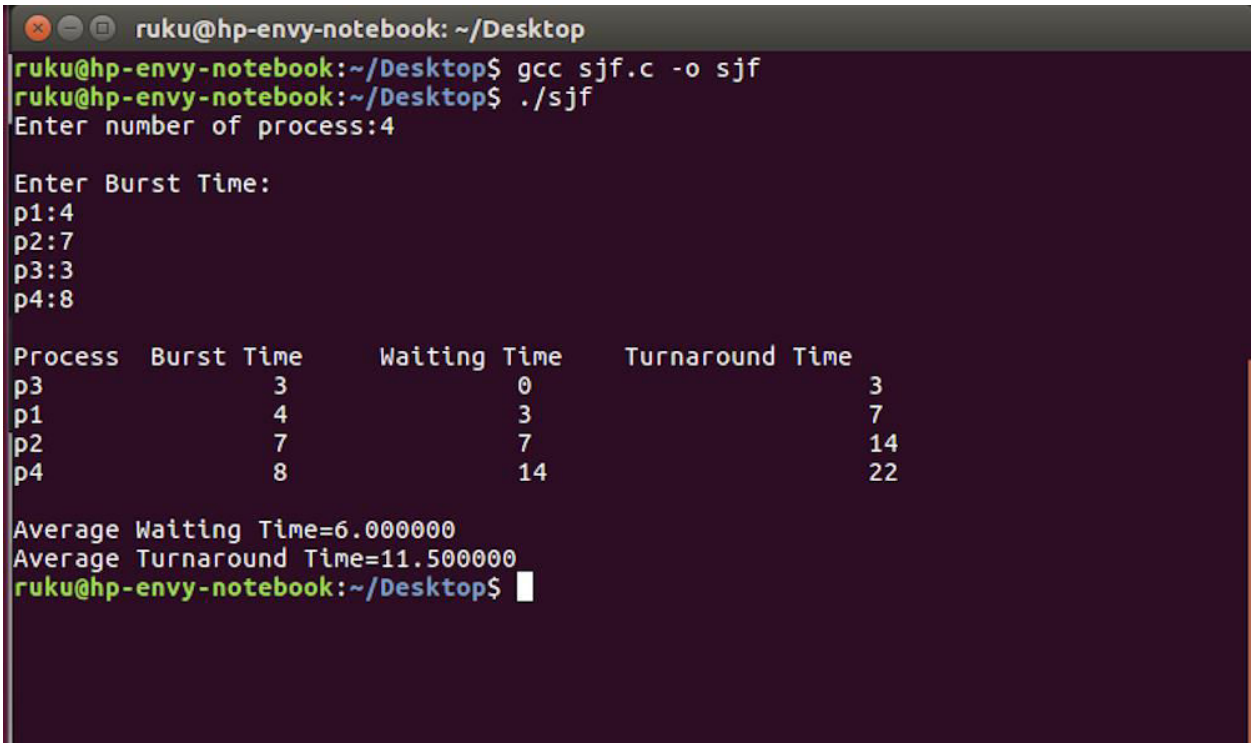
Implementation in C:

Code:

```
#include<stdio.h>

void 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;          //contains process number
   }

   //sorting burst time in ascending order using selection sort
   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;          //waiting time for first process will be zero

   //calculate waiting time
   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;      //average waiting time
```

```
    total=0;

    printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];     //calculate turnaround time
        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;     //average turnaround time
    printf("\n\nAverage Waiting Time=%f",avg_wt);
    printf("\nAverage Turnaround Time=%f\n",avg_tat);
}
```

Output:



Discussion:

From the implementation of this algorithm Shortest remaining time is the preemptive version of the SJN algorithm.The processor is allocated to the job closest tocompletion but it can be preemptive by a newer ready job with shorter time to completion.Impossible to implement in interactive system where required cpu time is not known.It is often used in batch environments where short jobs need to give preference.