Mawlana Bhashani Science and Technology University

Lab-Report

Report No: 8

Course code: ICT-3110

Course title: Operating Systems Lab

Date of Performance:

Date of Submission: 12/09/2020

Submitted by

Name: Mahfuza Talukdar

ID:IT-18009

3rd year 1st semester

Session: 2017-2018

Dept. of ICT

MBSTU.

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

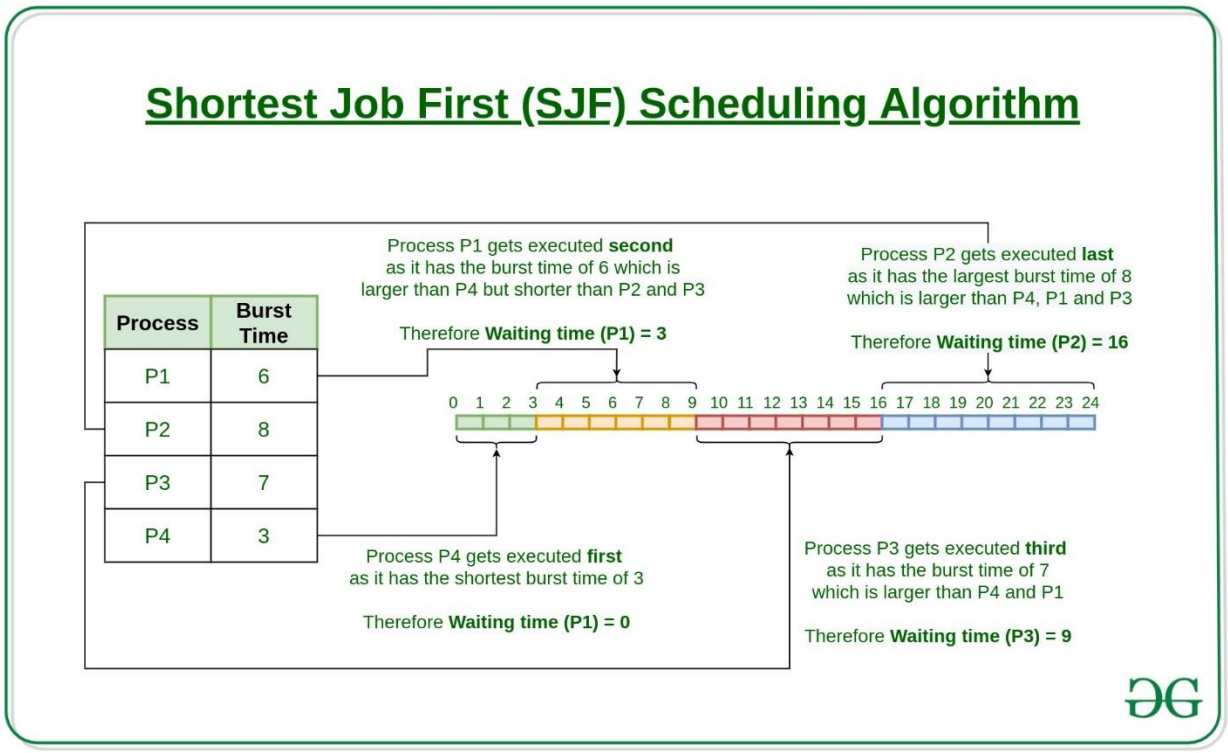
Experiment No : 8

Experiment Name : Implementation of SJF Scheduling Algorithm.

Theory :

Shortest job first (SJF) or shortest job next, is a scheduling policy that selects the waiting process with the smallest execution time to execute next. SJN is a non-preemptive algorithm.

- Shortest Job first has the advantage of having a minimum average waiting time among all scheduling algorithms.
- It is a Greedy Algorithm.
- It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of ageing.
- It is practically infeasible as Operating System may not know burst time and therefore may not sort them. While it is not possible to predict execution time, several methods can be used to estimate the execution time for a job, such as a weighted average of previous execution times. SJF can be used in specialized environments where accurate estimates of running time are available.



Corresponding Code:

Code for non-preemptive 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 processes :");
   scanf("%d",&n);

   printf("\nEnter Burst Time: ");
   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("\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 processes :5

Enter Burst Time: p1:4
p2:3
p3:7
p4:1
p5:2

Process     Burst Time          Waiting Time    Turnaround Time
p4                1                 0                  1
p5                2                 1                  3
p2                3                 3                  6
p1                4                 6                  10
p3                7                 10                 17

Average Waiting Time=4.000000
Average Turnaround Time=7.400000

Process returned 0 (0x0)   execution time : 14.070 s
Press any key to continue.
```

Code for preemptive SJF Scheduling Algorithm :

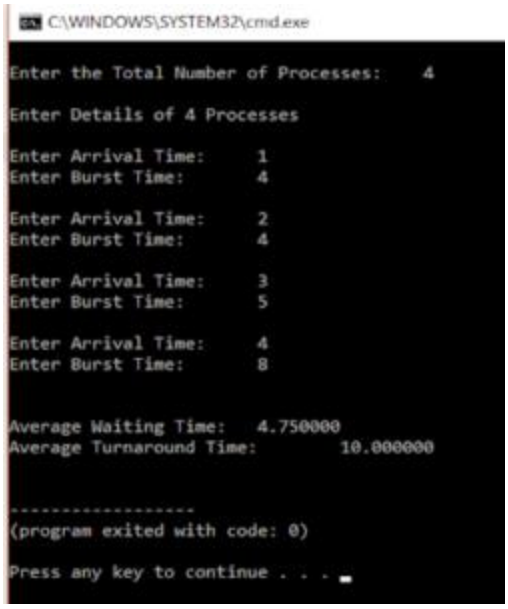
```
#include<stdio.h>
 int main()
{
  int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
  float avg_wt,avg_tat;
  printf("Enter number of processes :");
  scanf("%d",&n);
```

```
 printf("\nEnter Burst Time: ");
 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("\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 :



Discussion :

From this lab, we learn that how to implement non-preemptive and preemptive SJF (Smallest Job First) scheduling algorithm using C language. In future we can solve any problem of this algorithm.