



Mawlana Bhashani Science and Technology University

Lab-Report

Report No : 08
Experiment name : Implementation of SJF Scheduling Algorithm
Course code : ICT-3110
Course title : Operating System Lab.
Date of Performance :
Date of Submission :

Submitted by

Name: Iqbal Hossen
ID: IT-18041
3rd year 1st semester
Session: 2017-18
Dept. of ICT
MBSTU.

Submitted To

Nazrul Islam
Assistant Professor
Dept. of ICT
MBSTU.

i) What is SJF Scheduling Algorithm?

Shortest Job First(SJF):

Shortest Job First scheduling works on the process with the shortest **burst time** or **duration** first.

It is of two types:

1. Non Pre-emptive SJF.
2. Pre-emptive SJF.

1.Non Pre-emptive SJF:

Consider the below processes available in the ready queue for execution, with **arrival time** as 0 for all and given **burst times**.

Process	Burst time
P1	21
P2	3
P3	6
P4	2

In SJF scheduling shortest process is executed first. Hence the GANTT chart will be following:

P4	P2	P3	P1
0	2	5	11
			32

Process	A.T	B.T	W.T=(s.t-a.t) + (s.t-l.c.t)	T.A.T=B.T+W.T	C.T
P1	0	21	11	32	32

P2	0	3	2	5	5
P3	0	6	5	11	11
P4	0	2	0	2	2

$$\text{Average waiting time} = \frac{11+2+5+0}{4} = 4.5 \text{ ms}$$

$$\text{Average turn around time} = \frac{32+5+11+2}{4} = 12.5 \text{ ms}$$

2. Pre-emptive SJF:

In Preemptive Shortest Job First Scheduling, jobs are put into ready queue as they arrive, but as a process with **short burst time** arrives, the existing process is preempted or removed from execution, and the shorter job is executed first.

Process	Arrival time	Burst time
P1	0	21
P2	1	3
P3	2	6
P4	3	2

In SJF scheduling shortest process is executed first. Hence the GANTT chart will be following:

P1	P2	P2	P2	P4	P3	P1
----	----	----	----	----	----	----

0 1 2 3 4 6 12 32

Process	A.T	B.T	W.T=(s.t-a.t) + (s.t-l.c.t)	T.A.T=B.T+W.T	C.T
P1	0	21	11	32	32
P2	1	3	0	3	4

P3	2	6	4	10	12
P4	3	2	1	3	6

$$\text{Average waiting time} = \frac{11+0+4+1}{4} = 4.0 \text{ ms}$$

$$\text{Average turn around time} = \frac{32+3+10+3}{4} = 12.0 \text{ ms}$$

ii) Implementation of SJF algorithm in C

The implementation of Preemptive SJF scheduling algorithm in C is given below:

Code:

```
// SHORTEST JOB FIRST (Preemptive) Using C++
```

```
#include <iostream>
```

```
#include <algorithm>
```

```
#include <cstring>
```

```
using namespace std;
```

```
typedef struct proccess
```

```
{
```

```
    int at,bt,ct,ta,wt,btt;
```

```
    string pro_id;
```

```
} Schedule;
```

```
bool compare(Schedule a,Schedule b)
```

```
{
```

```
    return a.at<b.at;
```

```
}
```

```
bool compare2(Schedule a,Schedule b)
```

```
{
```

```
    return a.bt<b.bt;
```

```
}
```

```
int main()
```

```
{
```

```

Schedule pro[10];
int n,i,j,pcom;
double avg_wt,avg_tat,avg_ct,sum_wt=0,sum_tat=0,sum_ct=0;
cout<<"Enter the number of Process:";
cin>>n;

for(i=0; i<n; i++)
{
    cout<<"Enter the Process id, arrival time, burst time:";
    cin>>pro[i].pro_id>>pro[i].at>>pro[i].bt;
    pro[i].btt=pro[i].bt;
}

sort(pro,pro+n,compare);

i=0;
pcom=0;

while(pcom<n)
{
    for(j=0; j<n; j++)
    {
        if(pro[j].at>i)
            break;
    }

    sort(pro,pro+j,compare2);

    if(j>0)
    {
        for(j=0; j<n; j++)
        {
            if(pro[j].bt!=0)
                break;
        }
        if(pro[j].at>i)
        {
            i=pro[j].at;

```

```

    }
    pro[j].ct=i+1;
    pro[j].bt--;
}
i++;
pcom=0;
for(j=0; j<n; j++)
{
    if(pro[j].bt==0)
        pcom++;
}
}

```

```

cout<<"Process\tA.T\tB.T\tW.T\tT.A.T\tC.T\n";

```

```

for(i=0; i<n; i++)
{
    pro[i].ta=pro[i].ct-pro[i].at;
    pro[i].wt=pro[i].ta-pro[i].btt;

```

```

    sum_wt+=pro[i].wt;
    sum_tat+=pro[i].ta;
    sum_ct+=pro[i].ct;

```

```

    /*Printing the Process id, arrival time, burst time,
    completion time, turn around time, waiting time*/

```

```

cout<<pro[i].pro_id<<"\t"<<pro[i].at<<"\t"<<pro[i].btt<<"\t"<<pro[i].wt<
<"\t"<<pro[i].ta<<"\t"<<pro[i].ct;
    cout<<endl;
}

```

```

avg_wt=sum_wt/n;
avg_tat=sum_tat/n;
avg_ct=sum_ct/n;

```

```

cout<<"Average waiting time:"<<avg_wt<<endl;
cout<<"Average turn around time:"<<avg_tat<<endl;

```

```

    cout<<"Average completion time:"<<avg_ct<<endl;

    return 0;
}

```

Output:

```

Enter the number of Process:5
Enter the Process id, arrival time, burst time:p1 0 7
Enter the Process id, arrival time, burst time:p2 2 4
Enter the Process id, arrival time, burst time:p3 4 1
Enter the Process id, arrival time, burst time:p4 5 4
Enter the Process id, arrival time, burst time:p5 3 5
Process A.T    B.T    W.T    T.A.T    C.T
p3      4      1      0      1      5
p2      2      4      1      5      7
p4      5      4      2      6     11
p1      0      7      9     16     16
p5      3      5     13     18     21
Average waiting time:5
Average turn around time:9.2
Average completion time:12

Process returned 0 (0x0)    execution time : 41.148 s
Press any key to continue.

```