Experiment No :08

Experiment Name: Implementation of SJF Scheduling Algorithm.

Objectives:

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

❖ **SJF Scheduling Algorithm :**

SJF is a scheduling policy that selects the waiting process with the smallest execution time to execute next

Shortest Job First scheduling (SJF)

Aim : To write a c program to Implement the CPU scheduling algorithm Shortest job first (Non- Preemption)

Description:

To calculate the average waiting time in the shortest job first algorithm the sorting of the process based on their burst time in ascending order then calculate the waiting time of each process as the sum of the bursting times of all the process previous or before to that process

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

Source Code:

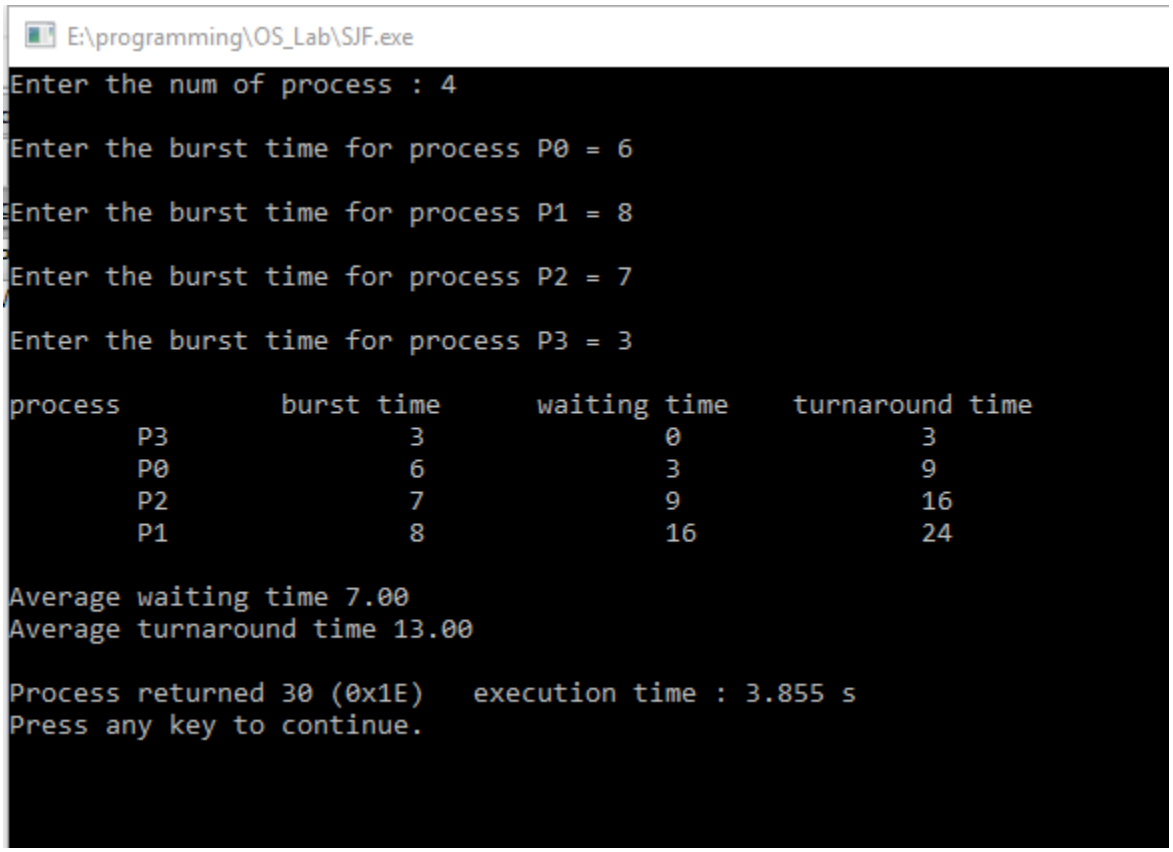
```
#include<stdio.h>
void main()
{
   int n,i,j,p[30],bt[30],temp,c,wt[30],tat[30];
   float awt,atat;
   printf("Enter the num of process : ");
   scanf("%d",&n);
   for(i=0; i<n; i++)
   {
      p[i]=i;
      printf("\nEnter the burst time for process P%d = ",i);
      scanf("%d",&bt[i]);
   }
   for(i=0; i<n; i++)
   {
      for(j=i+1; j<n; j++)
      {
         if(bt[i]>bt[j])
         {
            temp=bt[i];
            bt[i]=bt[j];
            bt[j]=temp;

            temp=p[i];
            p[i]=p[j];
            p[j]=temp;
         }
      }
   }
   printf("\nprocess \t burst time \t waiting time \t turnaround time\n");
```

```
    wt[0]=awt=0;
    tat[0]=atat=bt[0];
    for(i=1; i<n; i++)
    {
        wt[i]=wt[i-1]+bt[i-1];
        tat[i]=tat[i-1]+bt[i];

        awt=awt+wt[i];
        atat= atat+tat[i];
    }
    for(i=0; i<n; i++)
    {
        printf("\tP%d \t\t %d  \t\t %d \t\t %d\n",p[i],bt[i],wt[i],tat[i]);
    }
    awt=awt/n;
    atat=atat/n;
    printf("\nAverage waiting time %.2f\n",awt);
    printf("Average turnaround time %.2f\n",atat);
    return 0;
}
```

Output:



Result:

Thus the SJF program was executed and verified successfully.