NITHIN S
221IT085

# IT253 Lab Assignment 3

# Simulation of Preemptive Scheduling Algorithm

## Shortest Remaining Time First ( SRTF)

CODE

```c
#include <stdio.h>
#include <stdbool.h>

struct Process
{
    int pid;
    int AT;
    int BT;
    int BTF;
    int Priority;
    int CT;
    int TAT;
    int WT;
    bool visited;
};

int main()
{
    int n;
    printf("Enter the number of processes:");
    scanf("%d", &n);

    struct Process processes[n];

    printf("Enter the details of processes:\n");

    for (int i = 0; i < n; i++)
    {
        printf("Process %d:\n", i + 1);
        printf("Enter the PID: ");
        scanf("%d", &processes[i].pid);
        printf("Enter the Arrival Time: ");
        scanf("%d", &processes[i].AT);
        printf("Enter the Burst Time: ");
        scanf("%d", &processes[i].BT);
        processes[i].BTF = processes[i].BT;
        processes[i].visited = false;
        printf("\n");
    }
```

```
int i, j;
int minCT = 0;
for (j = 0; j < n;)
{
    int minBT, idx;
    minBT = idx = 1e9;
    for (int i = 0; i < n; i++)
    {
        if (processes[i].visited == false)
        {
            if (minCT >= processes[i].AT && minBT > processes[i].BT)
            {
                minBT = processes[i].BT;
                idx = i;
            }
            else if (minCT >= processes[i].AT && minBT == processes[i].BT)
            {
                if (processes[idx].AT > processes[i].AT)
                {
                    minBT = processes[i].BT;
                    idx = i;
                }
            }
        }
    }

    if (idx == 1e9)
    {
        minCT++;
    }
    else
    {
        minCT++;
        processes[idx].BT--;
        if (processes[idx].BT == 0)
        {
            j++;
            processes[idx].visited = true;
            processes[idx].CT = minCT;
        }
    }
}
int avg_TAT = 0;
int avg_WT = 0;
```

```
    for (int i = 0; i < n; i++)
    {
        processes[i].TAT = processes[i].CT - processes[i].AT;
        processes[i].WT = processes[i].TAT - processes[i].BTF;
        avg_TAT += processes[i].TAT;
        avg_WT += processes[i].WT;
    }

    printf("\nProcess\tAT\tBT\tCT\tTAT\tWT\n");

    for (int i = 0; i < n; i++)
    {
        printf("P%d\t%d\t%d\t%d\t%d\t%d\n", processes[i].pid, processes[i].AT, processes[i].BTF,
               processes[i].CT, processes[i].TAT, processes[i].WT);
    }

    printf("\nAverage Turn Around Time: %.2f", (float)avg_TAT / n);
    printf("\nAverage Waiting Time: %.2f\n", (float)avg_WT / n);

    return 0;
}
```

OUTPUT

```
nithin@nithin1729s:~/Codes/Sem4/IT253/Lab/Lab_4$ gcc SRTF.c
nithin@nithin1729s:~/Codes/Sem4/IT253/Lab/Lab_4$ ./a.out
Enter the number of processes:5
Enter the details of processes:
Process 1:
Enter the PID: 0
Enter the Arrival Time: 0
Enter the Burst Time: 4

Process 2:
Enter the PID: 1
Enter the Arrival Time: 1
Enter the Burst Time: 7

Process 3:
Enter the PID: 2
Enter the Arrival Time: 2
Enter the Burst Time: 1

Process 4:
Enter the PID: 3
Enter the Arrival Time: 3
Enter the Burst Time: 8

Process 5:
Enter the PID: 4
Enter the Arrival Time: 4
Enter the Burst Time: 5


Process AT      BT      CT      TAT     WT
P0      0       4       5       5       1
P1      1       7       17      16      9
P2      2       1       3       1       0
P3      3       8       25      22      14
P4      4       5       10      6       1

Average Turn Around Time: 10.00
Average Waiting Time: 5.00
nithin@nithin1729s:~/Codes/Sem4/IT253/Lab/Lab_4$
```

# Round Robin ( RR )

CODE

```c
#include <stdio.h>
#include <stdbool.h>

struct Process
{
    int pid;
    int AT;
    int BT;
    int BTF;
    int Priority;
    int CT;
    int TAT;
    int WT;
    bool visited;
};

int ready_queue[10000];

int main()
{
    int n;
    printf("Enter the number of processes:");
    scanf("%d", &n);

    struct Process processes[n];

    printf("Enter the details of processes:\n");
    int minAT=1e9;
    int initial_idx=-1;
    for (int i = 0; i < n; i++)
    {
        printf("Process %d:\n", i + 1);
        printf("Enter the PID: ");
        scanf("%d", &processes[i].pid);
        printf("Enter the Arrival Time: ");
        scanf("%d", &processes[i].AT);
        printf("Enter the Burst Time: ");
        scanf("%d", &processes[i].BT);
        processes[i].BTF = processes[i].BT;
        processes[i].visited = false;
        if(minAT>processes[i].AT)
        {
            minAT=processes[i].AT;
            initial_idx=i;
        }
        printf("\n");
    }
```

```
int TQ;
printf("Enter Time Quantum: ");
scanf("%d",&TQ);
int ans=0,curr_queue_idx=0,ready_queue_idx=0;
ready_queue[ready_queue_idx++]=initial_idx;
processes[initial_idx].visited=true;

for(int j=0;j<n;)
{
    int rem_time=processes[ready_queue[curr_queue_idx]].BT;
    int sub;
    if(rem_time>TQ)
    {
        ans+=TQ;
        processes[ready_queue[curr_queue_idx]].BT-=TQ;
        for(int i=0;i<n;i++)
        {
            if(processes[i].visited==false && processes[i].AT<=ans)
            {
                ready_queue[ready_queue_idx++]=i;
                processes[i].visited=true;
            }
        }
        ready_queue[ready_queue_idx++]=ready_queue[curr_queue_idx];
    }
    else
    {
        ans+=rem_time;
        processes[ready_queue[curr_queue_idx]].BT-=0;
        for(int i=0;i<n;i++)
        {
            if(processes[i].visited==false && processes[i].AT<=ans)
            {
                ready_queue[ready_queue_idx++]=i;
                processes[i].visited=true;
            }
        }
        processes[ready_queue[curr_queue_idx]].CT=ans;
        j++;
    }
    curr_queue_idx++;


}
```

```
    int avg_TAT = 0;
    int avg_WT = 0;
    for (int i = 0; i < n; i++)
    {
        processes[i].TAT = processes[i].CT - processes[i].AT;
        processes[i].WT = processes[i].TAT - processes[i].BTF;
        avg_TAT += processes[i].TAT;
        avg_WT += processes[i].WT;
    }

    printf("\nProcess\tAT\tBT\tCT\tTAT\tWT\n");

    for (int i = 0; i < n; i++)
    {
        printf("P%d\t%d\t%d\t%d\t%d\t%d\n", processes[i].pid, processes[i].AT, processes[i].BTF,
               processes[i].CT, processes[i].TAT, processes[i].WT);
    }

    printf("\nAverage Turn Around Time: %.2f", (float)avg_TAT / n);
    printf("\nAverage Waiting Time: %.2f\n", (float)avg_WT / n);

    return 0;
}
```

# OUTPUT

```
nithin@nithin1729s:~/Codes/Sem4/IT253/Lab/Lab_4$ gcc RR.c
nithin@nithin1729s:~/Codes/Sem4/IT253/Lab/Lab_4$ ./a.out
Enter the number of processes:5
Enter the details of processes:
Process 1:
Enter the PID: 0
Enter the Arrival Time: 0
Enter the Burst Time: 4

Process 2:
Enter the PID: 1
Enter the Arrival Time: 1
Enter the Burst Time: 7

Process 3:
Enter the PID: 2
Enter the Arrival Time: 2
Enter the Burst Time: 1

Process 4:
Enter the PID: 3
Enter the Arrival Time: 3
Enter the Burst Time: 8

Process 5:
Enter the PID: 4
Enter the Arrival Time: 4
Enter the Burst Time: 5

Enter Time Quantum: 2

Process AT      BT      CT      TAT     WT
P0      0       4       7       7       3
P1      1       7       23      22      15
P2      2       1       5       3       2
P3      3       8       25      22      14
P4      4       5       22      18      13

Average Turn Around Time: 14.40
Average Waiting Time: 9.40
nithin@nithin1729s:~/Codes/Sem4/IT253/Lab/Lab_4$
```