CS331

S5 SEMESTER

System Software Lab

Github : [ceccs18c59/cs331: System Software Lab (github.com)]

Experiment No 1

Write a C program to simulate the non-preemptive CPU scheduling algorithms for finding turnaround time and waiting time.

1. First Come First Serve (FCFS)

2. Shortest Job First (SJF)

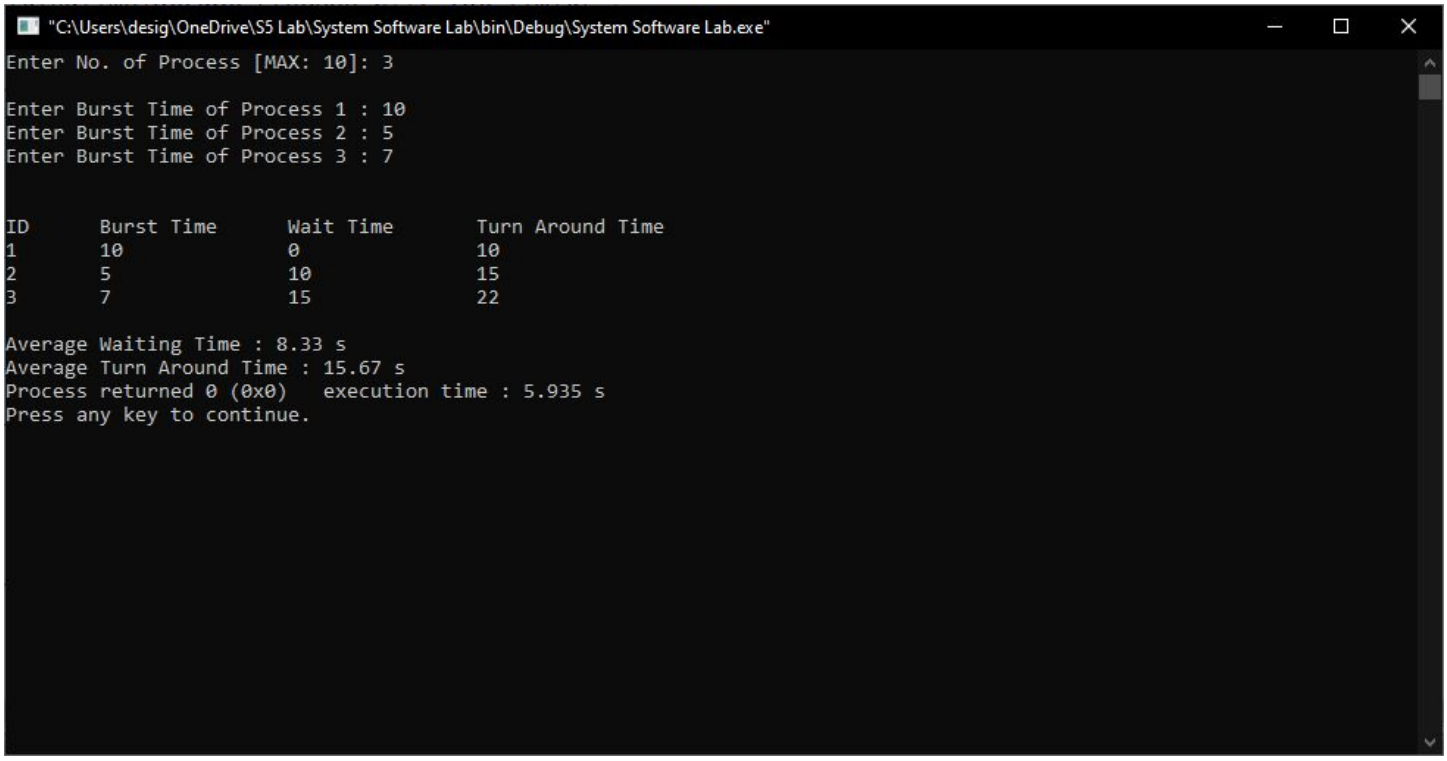
1. First Come First Serve (FCFS)

Program

```
#include <stdio.h>
#include <conio.h>

const int processLimit = 10;
float avgWaitTime;
float avgTurnArndTime;

struct Process
{
    int id;
    int burstTime;
    int waitTime;
    int turnArndTime;
};

int calculateWaitingTime(struct Process p[], int limit)
{

    int totalWaitTime = 0;
    int totalTurnArndTime = 0;
    p[0].waitTime = 0;
    for (int i = 1; i < limit; i++)
    {
        p[i].waitTime = p[i - 1].waitTime + p[i - 1].burstTime;
    }
    for (int i = 0; i < limit; i++)
    {
        p[i].turnArndTime = p[i].waitTime + p[i].burstTime;
    }
    for (int i = 0; i < limit; i++)
    {
        totalWaitTime = totalWaitTime + p[i].waitTime;
```

```
        totalTurnArndTime = totalTurnArndTime + p[i].turnArndTime;
    }
    avgTurnArndTime = (float)totalTurnArndTime / limit;
    avgWaitTime = (float)totalWaitTime / limit;

    return 0;
};

int display(struct Process p[], int limit)
{
    printf("\n\nID\tBurst Time\tWait Time\tTurn Around Time");
    for (int i = 0; i < limit; i++)
    {
        printf("\n%d\t%d\t\t%d\t\t%d", p[i].id, p[i].burstTime, p[i].waitTime,
p[i].turnArndTime);
    }
    return 0;
};

int main()
{
    int limit = 0;
    printf("Enter No. of Process [MAX: %d]: ", processLimit);
    scanf("%d", &limit);
    struct Process p[processLimit];
    printf("\n");

    for (int i = 0; i < limit; i++)
    {
        printf("Enter Burst Time of Process %d : ", i + 1);
        scanf("%d", &p[i].burstTime);
        p[i].id = i + 1;
    }

    calculateWaitingTime(p, limit);
    display(p, limit);
    printf("\n\nAverage Waiting Time : %0.2f s\n", avgWaitTime);
    printf("Average Turn Around Time : %0.2f s", avgTurnArndTime);
    getch();
};
```

Output



```
"C:\Users\desig\OneDrive\S5 Lab\System Software Lab\bin\Debug\System Software Lab.exe"
Enter No. of Process [MAX: 10]: 3

Enter Burst Time of Process 1 : 10
Enter Burst Time of Process 2 : 5
Enter Burst Time of Process 3 : 7


ID      Burst Time      Wait Time       Turn Around Time
1       10              0               10
2       5               10              15
3       7               15              22

Average Waiting Time : 8.33 s
Average Turn Around Time : 15.67 s
Process returned 0 (0x0)   execution time : 5.935 s
Press any key to continue.
```

2. Shortest Job First (FCFS)

Program

```
#include <stdio.h>
#include <conio.h>

const int processLimit = 10;
float avgWaitTime;
float avgTurnArndTime;

struct Process
{
    int id;
    int burstTime;
    int waitTime;
    int turnArndTime;
};

int sortProcesses(struct Process p[], int limit)
```

```
{
    int pos;
    struct Process temp;
    for (int i = 0; i < limit; i++)
    {
        pos = i;
        for (int j = i + 1; j < limit; j++)
        {
            if (p[j].burstTime < p[pos].burstTime)
                pos = j;
        }
        temp = p[i];
        p[i] = p[pos];
        p[pos] = temp;
    }
    return 0;
};

int calculateWaitingTime(struct Process p[], int limit)
{

    int totalWaitTime = 0;
    int totalTurnArndTime = 0;
    p[0].waitTime = 0;
    for (int i = 1; i < limit; i++)
    {
        p[i].waitTime = p[i - 1].waitTime + p[i - 1].burstTime;
    }
    for (int i = 0; i < limit; i++)
    {
        p[i].turnArndTime = p[i].waitTime + p[i].burstTime;
    }
    for (int i = 0; i < limit; i++)
    {
        totalWaitTime = totalWaitTime + p[i].waitTime;
        totalTurnArndTime = totalTurnArndTime + p[i].turnArndTime;
    }
    avgTurnArndTime = (float)totalTurnArndTime / limit;
    avgWaitTime = (float)totalWaitTime / limit;

    return 0;
};

int display(struct Process p[], int limit)
{
    printf("\n\nID\tBurst Time\tWait Time\tTurn Around Time");
    for (int i = 0; i < limit; i++)
    {
        printf("\n%d\t%d\t\t%d\t\t%d", p[i].id, p[i].burstTime, p[i].waitTime,
p[i].turnArndTime);
    }
    return 0;
};

int main()
{
```

```
    int limit = 0;
    printf("Enter No. of Process [MAX: %d]: ", processLimit);
    scanf("%d", &limit);
    struct Process p[processLimit];
    printf("\n");

    for (int i = 0; i < limit; i++)
    {
        printf("Enter Burst Time of Process %d : ", i + 1);
        scanf("%d", &p[i].burstTime);
        p[i].id = i + 1;
    }

    sortProcesses(p, limit);
    calculateWaitingTime(p, limit);
    display(p, limit);
    printf("\n\nAverage Waiting Time : %0.2f s\n", avgWaitTime);
    printf("Average Turn Around Time : %0.2f s", avgTurnArndTime);
    getch();
};
```

Output