

System Software Lab

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

Experiment No 1

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

1. Round Robin (RR)
2. Priority

1. Round Robin (RR)

Program

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

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

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

int calculateWaitingTime(struct Process p[], int limit, int timeSlice)
{
    int totalWaitTime = 0;
    int totalTurnArndTime = 0;
    int t = 0;
    bool done = false;

    while (true)
    {
        done = true;
        for (int i = 0; i < limit; i++)
        {
            if (p[i].remBurstTime > 0)
            {
```

```

        done = false;

        if (p[i].remBurstTime > timeSlice)
        {
            t += timeSlice;
            p[i].remBurstTime -= timeSlice;
        }
        else
        {
            t = t + p[i].remBurstTime;
            p[i].waitTime = t - p[i].burstTime;
            p[i].remBurstTime = 0;
        }
    }

    if (done == true)
        break;
}

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

return t;
};

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

int main()
{
    int limit = 0, timeSlice, time;
    printf("Enter No. of Process [MAX: %d]: ", processLimit);
    scanf("%d", &limit);
    printf("Enter time slice(Quantum) of CPU: ", timeSlice);
    scanf("%d", &timeSlice);

    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].remBurstTime = p[i].burstTime;
    p[i].id = i + 1;
    p[i].waitTime = 0;
    p[i].turnArndTime = 0;
}

time = calculateWaitingTime(p, limit, timeSlice);
display(p, limit, time);
printf("\n\nAverage Waiting Time : %0.2f s\n", avgWaitTime);
printf("Average Turn Around Time : %0.2f s", avgTurnArndTime);
getch();
return 0;
};

```

Output

```

C:\Users\Thejus\Desktop\Lab\cs331\Experiment 2\rr.exe
Enter No. of Process [MAX: 10]: 5
Enter time slice(Quantum) of CPU: 2

Enter Burst Time of Process 1 : 4
Enter Burst Time of Process 2 : 3
Enter Burst Time of Process 3 : 8
Enter Burst Time of Process 4 : 10
Enter Burst Time of Process 5 : 2

ID      Burst Time      Wait Time      Turn Around Time
1        4             8             12
2        3            10            13
3        8            15            23
4       10            17            27
5         2             8             10

Total Time Taken: 27

Average Waiting Time : 11.60 s
Average Turn Around Time : 17.00 s

```

2. Priority

Program

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

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

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

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].priority < p[pos].priority)
                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, timeSlice, time;
    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 & Priority %d : ", i + 1);
        scanf("%d %d", &p[i].burstTime, &p[i].priority);
        p[i].id = i + 1;
        p[i].waitTime = 0;
        p[i].turnArndTime = 0;
    }
    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();
    return 0;
};

```

Output

C:\Users\Thejus\Desktop\Lab\cs331\Experiment 2\priority.exe

Enter No. of Process [MAX: 10]: 4

Enter Burst Time of Process & Priority 1 : 6 3

Enter Burst Time of Process & Priority 2 : 2 2

Enter Burst Time of Process & Priority 3 : 14 1

Enter Burst Time of Process & Priority 4 : 6 4

ID	Burst Time	Wait Time	Turn Around Time
3	14	0	14
2	2	14	16
1	6	16	22
4	6	22	28

Average Waiting Time : 13.00 s

Average Turn Around Time : 20.00 s