Mawlana Bhashani Science And Technology University

Lab-Report

Report No : 10

Course Code : ICT-3110

Course Title : Operating System Lab.

Date of Performance :

Date of Submission : 25/09/2020

Submitted By:

Name:Md Abdur Rahim

ID:IT-18024

3rd Year 1st Semester

Session : 2017-18

Dept. of ICT

MBSTU

Submitted To:

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU

Experiment No: 10

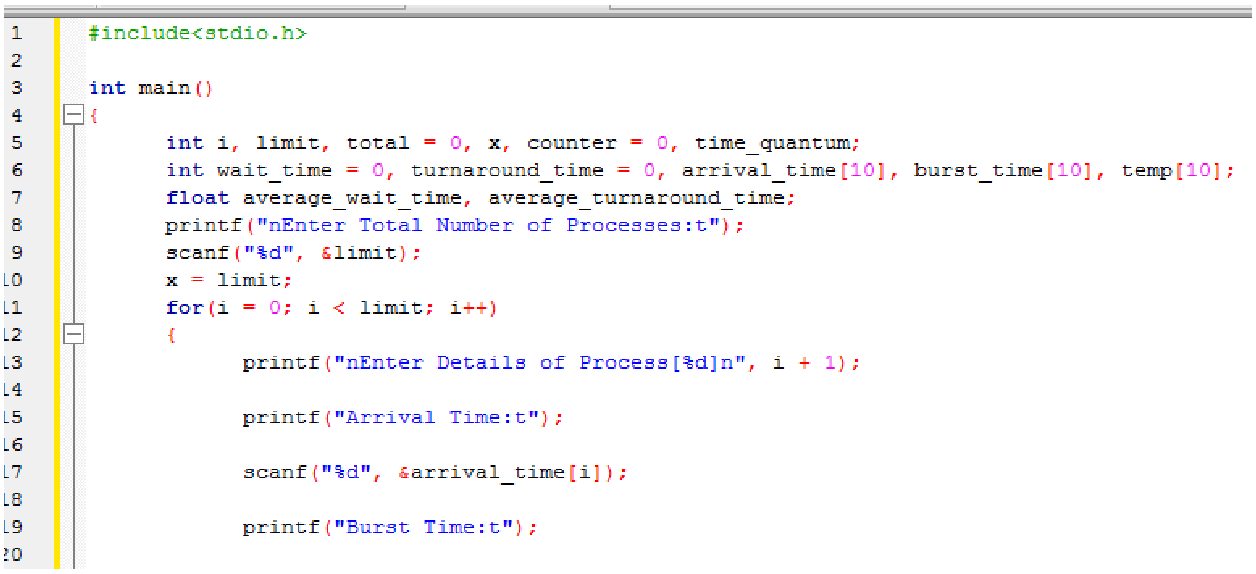
Experiment Name: Implementation of Round Robin Scheduling Algorithm.

Round Robin Scheduling is a scheduling algorithm used by the system to schedule CPU utilization. This is a preemptive algorithm . There exist a fixed time slice associated with each request called the quantum. The job scheduler saves the progress of the job that is being executed currently and moves to the next job present in the queue when a particular process is executed for a given time quantum.

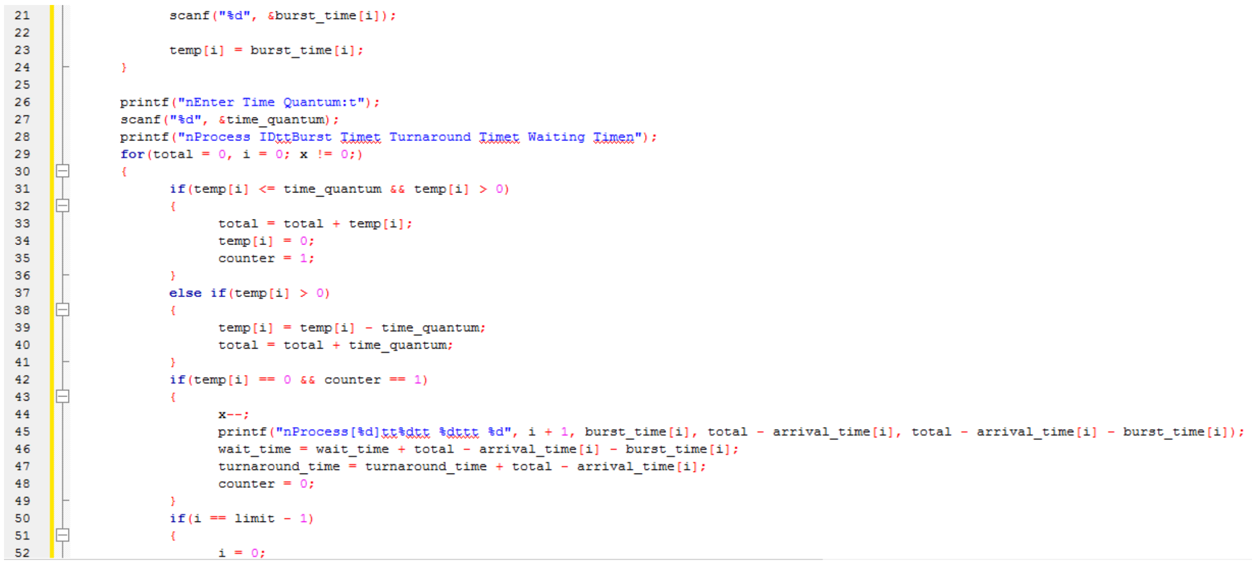
Algorithm:

- We first have a queue where the processes are arranged in first come first serve order.
- A quantum value is allocated to execute each process.
- The first process is executed until the end of the quantum value. After this, an interrupt is generated and the state is saved.
- The CPU then moves to the next process and the same method is followed.
- Same steps are repeated till all the processes are over.

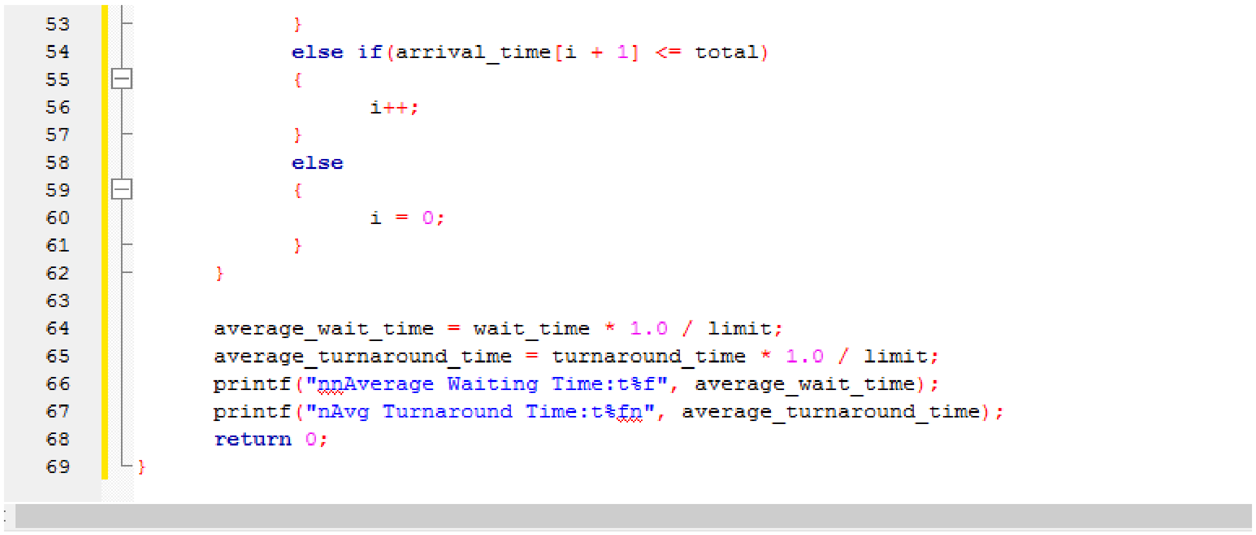
Code Implementation :



```
#include<stdio.h>

int main()
{
      int i, limit, total = 0, x, counter = 0, time_quantum;
      int wait_time = 0, turnaround_time = 0, arrival_time[10], burst_time[10], temp[10];
      float average_wait_time, average_turnaround_time;
      printf("nEnter Total Number of Processes:t");
      scanf("%d", &limit);
      x = limit;
      for(i = 0; i < limit; i++)
      {
            printf("nEnter Details of Process[%d]n", i + 1);

            printf("Arrival Time:t");

            scanf("%d", &arrival_time[i]);

            printf("Burst Time:t");

```

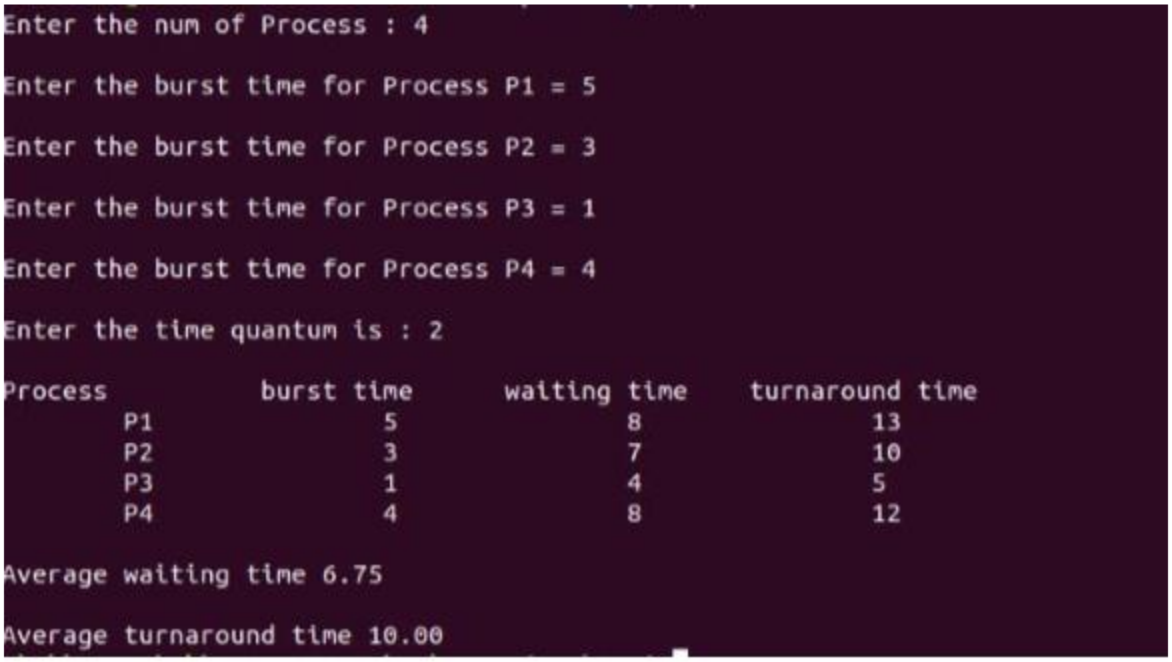


```c
        scanf("%d", &burst_time[i]);

        temp[i] = burst_time[i];
    }

    printf("nEnter Time Quantum:t");
    scanf("%d", &time_quantum);
    printf("nProcess IDttBurst Timet Turnaround Timet Waiting Timen");
    for(total = 0, i = 0; x != 0;)
    {
        if(temp[i] <= time_quantum && temp[i] > 0)
        {
            total = total + temp[i];
            temp[i] = 0;
            counter = 1;
        }
        else if(temp[i] > 0)
        {
            temp[i] = temp[i] - time_quantum;
            total = total + time_quantum;
        }
        if(temp[i] == 0 && counter == 1)
        {
            x--;
            printf("nProcess[%d]tt%dtt %dttt %d", i + 1, burst_time[i], total - arrival_time[i], total - arrival_time[i] - burst_time[i]);
            wait_time = wait_time + total - arrival_time[i] - burst_time[i];
            turnaround_time = turnaround_time + total - arrival_time[i];
            counter = 0;
        }
        if(i == limit - 1)
        {
            i = 0;
```



```c
        }
        else if(arrival_time[i + 1] <= total)
        {
            i++;
        }
        else
        {
            i = 0;
        }
    }

    average_wait_time = wait_time * 1.0 / limit;
    average_turnaround_time = turnaround_time * 1.0 / limit;
    printf("nnAverage Waiting Time:t%f", average_wait_time);
    printf("nAvg Turnaround Time:t%fn", average_turnaround_time);
    return 0;
}
```

Output:



```
Enter the num of Process : 4

Enter the burst time for Process P1 = 5

Enter the burst time for Process P2 = 3

Enter the burst time for Process P3 = 1

Enter the burst time for Process P4 = 4

Enter the time quantum is : 2

Process         burst time      waiting time    turnaround time
       P1               5               8               13
       P2               3               7               10
       P3               1               4               5
       P4               4               8               12

Average waiting time 6.75

Average turnaround time 10.00
```

Discussion:

In the above code, we ask the user to enter the number of processes and arrival time and burst time for each process. We then calculate the waiting time and the turn around time using the round-robin algorithm.