MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY

Santosh,Tangail-1902

LAB REPORT

Department of   : Information & Communication Technology

Lab Report No   : 10

Lab Report On   : **Implementation of Round Robin Scheduling Algorithm**

Course Title   : Operating Systems Lab

Course Code   :   ICT - 3110

Submitted by,	Submitted to,
Name : Zahid Hasan Chowdhury	Nazrul Islam
Student ID : IT-18017	Assistant Professor
Session : 2017-18	Dept. of ICT, MBSTU
Year: 3rd Semester : 1st	
Dept. of ICT MBSTU	

Experiment No : 10

Experiment Name : Implementation of Round Robin scheduling algorithm

Theory :

The name of this algorithm comes from the round-robin principle, where each person gets an equal share of something in turns. It is the oldest, simplest scheduling algorithm, which is mostly used for multitasking.

In Round-robin scheduling, each ready task runs turn by turn only in a cyclic queue for a limited time slice. This algorithm also offers starvation free execution of processes.

- Round robin is a pre-emptive algorithm
- The CPU is shifted to the next process after fixed interval time, which is called time quantum/time slice.
- The process that is preempted is added to the end of the queue.
- Round robin is a hybrid model which is clock-driven
- Time slice should be minimum, which is assigned for a specific task that needs to be processed. However, it may differ OS to OS.
- It is a real time algorithm which responds to the event within a specific time limit.
- Round robin is one of the oldest, fairest, and easiest algorithm.
- Widely used scheduling method in traditional OS.

Working Process :

```cpp
// C++ program for implementation of RR
//scheduling
#include<iostream>
using namespace std;

void findWaitingTime(int processes[], int n,
                     int bt[], int wt[], int quantum)
{
    int rem_bt[n];
    for (int i = 0 ; i < n ; i++)
            rem_bt[i] = bt[i];

    int t = 0;

    while (1)
    {
            bool done = true;
```

```cpp
        for (int i = 0 ; i < n; i++)
        {
                if (rem_bt[i] > 0)
                {
                        done = false;

                        if (rem_bt[i] > quantum)
                        {
                                t += quantum;
                                rem_bt[i] -= quantum;
                        }
                        else
                        {
                                t = t + rem_bt[i];
                                wt[i] = t - bt[i];
                                rem_bt[i] = 0;
                        }
                }
        }
        if (done == true)
        break;
    }
}
void findTurnAroundTime(int processes[], int n,int bt[], int wt[], int tat[])
{
        for (int i = 0; i < n ; i++)
                tat[i] = bt[i] + wt[i];
}
void findavgTime(int processes[], int n, int bt[],int quantum)
{
        int wt[n], tat[n], total_wt = 0, total_tat = 0;
        findWaitingTime(processes, n, bt, wt, quantum);
        findTurnAroundTime(processes, n, bt, wt, tat);
        cout << "Processes "<< " Burst time "
                << " Waiting time " << " Turn around
        time\n"; for (int i=0; i<n; i++)
        {
                total_wt =  total_wt  +  wt[i];
                total_tat = total_tat + tat[i];
                cout << " " << i+1 << "\t\t" << bt[i] <<"\t "
                        << wt[i] <<"\t\t " << tat[i] <<endl;
        }

        cout << "Average waiting time = "
```

```
                << (float)total_wt / (float)n; cout
        << "\nAverage turn around time = "
                << (float)total_tat / (float)n;
}
int main()
{
        // process id's
        int processes[] = { 1, 2, 3};
        int n = sizeof processes / sizeof processes[0];

        // Burst time of all processes
        int burst_time[] = {10, 5, 8};

        // Time quantum
        int quantum = 2;
        findavgTime(processes, n, burst_time,
        quantum); return 0;
}
```

Output :

```
Processes   Burst time   Waiting time   Turn around time
1              10           13               23
2              5            10               15
3              8            13               21
Average waiting time = 12
Average turn around time = 19.6667
Process returned 0 (0x0)    execution time : 0.021 s
Press any key to continue.
```

Discussion :

This lab helps to learn Round Robin scheduling algorithm. We have implemented this algorithm using C language. Program worked fine and result was correct.