Mawlana Bhashani Science and Technology University

Lab Report

Report No: 10

Course Code: ICT – 3110

Course title: Operating Systems Lab

Date of Performance:

Date of Submission: 12/09/2020

Submitted by

Name: Pritom Saha

ID: IT – 17010

3rd year 1st semester

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.

Theory:

It is a CPU scheduling algorithm. In this algorithm the CPU is shifted to the next process after fixed interval time which is called time slice. It is preemptive as processes are assigned CPU only for a fixed slice of time at most. It has some disadvantage too. Time consuming scheduling only works for the small quantums.

There are some topics which are used in this algorithm. Those are briefly explained below –

1. **Completion time: time at which process completes its execution.**
2. **Turnaround time: turnaround time = Completion time – Arrival time.**
3. **Waiting time = Waiting time = Turnaround time – Burst time.**

Working process:

1. **First of all, we have to create an array rem_burst_time[] to keep track of remaining burst time of process. This array is initially a copy of burst time array.**
2. **Then we have to create another array waiting_time[] to store waiting times of processes. Initialize the array as 0.**
3. **Initially t = 0.**
4. **Then we have to keep traversing the all processes while all processes are not done.**

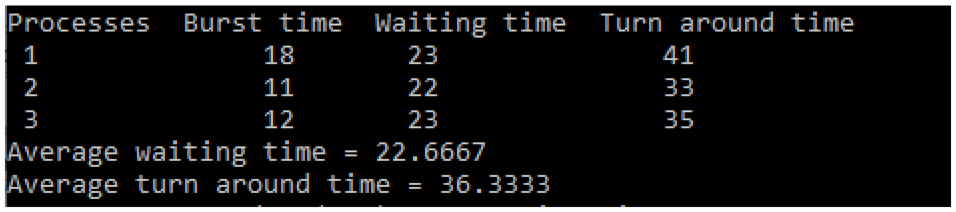
```
If rem_burst_time[i] > quantum
Then (i) t += quantum;
     (ii) rem_burst_time[i] -= quantum;
else
     (i)    t += rem_burst_time[i];
     (ii)   waiting_time[i] = t – burst_time[i]
     (iii)  rem_burst_time[i] = 0;
```

Code:

```
#include<bits/stdc++.h>
using namespace std;
signed main() {
    ios :: sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    int n = 3;
    int process[n] = {1, 2, 3};
    int burst_time[n] = {18, 11, 12};
    int quantum = 2;
    int waiting_time[n], turn_around_time[n];
    int total_waiting_time = 0;
    int total_turn_around_time = 0;
    int rem_burst_time[n];
    for (int i = 0; i < n; i++) {
        rem_burst_time[i] = burst_time[i];
    }
    int cnt = 0;
    while(true) {
        bool done = true;
        for (int i = 0; i < n; i++) {
            if (rem_burst_time[i] > 0) {
                done = false;
                if (rem_burst_time[i] > quantum) {
                    cnt += quantum;
                    rem_burst_time[i] -= quantum;
                } else {
                    cnt += rem_burst_time[i];
                    waiting_time[i] = cnt - burst_time[i];
                    rem_burst_time[i] = 0;
                }
```

```
                }
            }
            if (done) {
                break;
            }
        }
        for (int i = 0; i < n; i++) {
            turn_around_time[i] = burst_time[i] + waiting_time[i];
        }
        cout << "Processes " << " Burst time " << " Waiting time " << " Turn around 
    time" << endl;
        for (int i = 0; i < n; i++) {
            total_waiting_time += waiting_time[i];
            total_turn_around_time += turn_around_time[i];
            cout << " " << i + 1 << "\t\t" << burst_time[i] << "\t " << waiting_time[i] 
    <<"\t\t " << turn_around_time[i] <<endl;
        }
        cout << "Average waiting time = " << (double)total_waiting_time / (double)n;
        cout << "\nAverage turn around time = "<< (double)total_turn_around_time 
    / (double)n;
        return 0;
    }
```

Output:



```
Processes  Burst time  Waiting time  Turn around time
 1              18         23               41
 2              11         22               33
 3              12         23               35
Average waiting time = 22.6667
Average turn around time = 36.3333
```

Discussion:

The biggest advantage of the Round Robin Scheduling Algorithm is that if the total number of processes on the run queue, then the worst – case response time for the same process. This algorithm spends more time on context switching.