Mawlana Bhashani Science and Technology University

Lab Report

Report No: 07

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: 07

Experiment name: Implementation of FCFS scheduling algorithm.

Theory:

FCFS stands for first come first served. It is an algorithm which processes queues in such a way they arrive in a ready queue. We took the arrival time for all processes is equal to 0.

The required time for process to complete its execution is called completion time.

The difference between completion time and arrival time is called turnaround time.

The difference between turnaround time and burst time is called waiting time.

Working process:

1. **First of all we will take the total process number along their burst time.**
2. **Then, we will compute the waiting time for all process number. For first process the waiting time will be always 0 so wating_time[0] = 0;**
3. **By using dynamic programming we will calculate the waiting time of other process.**
   waiting_time[i] = waiting_time[i – 1] + burst_time[i – 1];
4. **Then, we will calculate turnaround time. Turnaround time = waiting time + burst time.**

5. Then, we will calculate average waiting time. Average waiting time = total waiting time / number of process.
6. Finally we will calculate average turnaround time. Average turnaround time = total turnaround time / number of process.

Code:

```
#include<bits/stdc++.h>
using namespace std;

signed main() {
    ios :: sync_with_stdio(false);
cin.tie(0); cout.tie(0);
int total_process = 4;
int proces[4] = {1, 2, 3, 4};
int burst_time[4] = {17, 12, 4, 7};
int waiting_time[4];
waiting_time[0] = 0;
for (int i = 1; i < total_process; i++) {
    waiting_time[i] = waiting_time[i - 1] + burst_time[i - 1];
}
int turn_around_time[4];
```

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

Output:

```
Porcesses   Burst time   Waiting time   Turn around time
   1            17              0              17
   2            12              17             29
   3            4               29             33
   4            7               33             40
Average waiting time = 19.75
Average turn around time = 29.75
```

<u>Discussion</u>: The algorithm is non – preemptive. It works on O(n) time complexity.