**GANDAKI COLLEGE OF ENGINEERING AND SCIENCE**

LAB 1: - IMPLEMENT FCFS SCHEDULING ALGORITHM

**SUBMITTED BY: SUBMITTED TO: -**

**Shivaji Chalise PRATIKSHYA SHRESTHA**

**BE2019SE685 Teacher**

**OBJECTIVE:** TO IMPLEMENT FCFS SCHEDULING WITH/WITHOUT ARRIVAL TIME

**THEORY:** FCFS stands for First Come First Serve. It is an operating system scheduling algorithm that automatically executes queued requests and processes in order of their arrival. It is the easiest and simplest CPU scheduling algorithm. In this type of algorithm, processes which requests the CPU first get the CPU allocation first. This is managed with a FIFO (First In First Out) queue. FCFS is a non-preemptive scheduling algorithm as a process holds the CPU until it either terminates or performs I/O. Thus, if a longer job has been assigned to the CPU then many shorter jobs after it will have to wait. This algorithm is used in most batch operating systems.

**SOURCE CODE:**

FCFS scheduling with arrival time

#include <iostream>

using namespace std;

void calculateWaitingTime(int arrivalTime[], int burstTime[], int N) {

int waitingTime[N];

// waiting time for first process is 0

waitingTime[0] = 0;

cout << "PN\t\tAT\t\t"

<< "BT\t\tWT\n\n";

cout << "1"

<< "\t\t" << arrivalTime[0] << "\t\t" << burstTime[0] << "\t\t"

<< waitingTime[0] << endl;

// calculating waiting time for each process from the given formula

for (int i = 1; i < 5; i++) {

waitingTime[i] =

(arrivalTime[i - 1] + burstTime[i - 1] + waitingTime[i - 1]) -

arrivalTime[i];

cout << i + 1 << "\t\t" << arrivalTime[i] << "\t\t" << burstTime[i]

<< "\t\t" << waitingTime[i] << endl;

}

float average;

float sum = 0;

for (int i = 0; i < 5; i++) {

sum = sum + waitingTime[i];

}

average = sum / 5;

cout << "\nAverage waiting time = " << average;

}

int main() {

int processes = 5;

int arrivalTime[] = {0, 1, 2, 3, 4};

int burstTime[] = {4, 3, 1, 2, 5};

calculateWaitingTime(arrivalTime, burstTime, processes);

return 0;

}

Output:

PN AT BT WT

1 0 4 0

2 1 3 3

3 2 1 5

4 3 2 5

5 4 5 6

Average waiting time = 3.8

FCFS scheduling without arrival time

#include <iostream>

using namespace std;

void findWaitingTime(int processes[], int n, int bt[], int wt[]) {

wt[0] = 0;

for (int i = 1; i < n; i++)

wt[i] = bt[i - 1] + wt[i - 1];

}

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 wt[n], tat[n], total\_wt = 0, total\_tat = 0;

findWaitingTime(processes, n, bt, wt);

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() {

int processes[] = {1, 2, 3};

int n = sizeof processes / sizeof processes[0];

int burst\_time[] = {10, 5, 8};

findavgTime(processes, n, burst\_time);

return 0;

}

```

Output

Processes Burst time Waiting time Turn around time

1 10 0 10

2 5 10 15

3 8 15 23

Average waiting time = 8.33333

Average turn around time = 16

**CONCLUSION:** Finally we understood and learned to implement FCFS Scheduling Algorithm.