...for a complete human being

Texas International College

Lab No: 4 **Date: 2082/**

Title: Write a program to calculate the average turnaround time and waiting time for user input process parameters using FCFS process scheduling algorithm.

First Come First Serve (FCFS) is the simplest CPU scheduling algorithm. In this method, the process that arrives first in the ready queue is executed first. It follows the FIFO (First In First Out) principle, similar to a queue in real life.

Algorithm:

Step 1: Start with the list of processes and note their arrival time and burst time.

Step 2: Sort all processes in the order of their arrival time.

Step 3: Assign the CPU to the process that arrived first, then to the next, and so on.

Step 4: Calculate waiting time and turnaround time for each process.

Step 5: Compute the average waiting time and average turnaround time.

Step 6: Stop.

Language: C++

IDE: VS Code

Code:

```
#include <iostream>
#include <vector>
using namespace std;


struct Job {
    int pid;        // process id
    int bt;         // burst time
    int at;         // arrival time
    int ct;         // completion time
    int tat;        // turnaround time
    int wt;         // waiting time
};
int main() {
    int num;
    cout << "Enter number of processes: ";
    cin >> num;
    vector<Job> tasks(num);
    // All arrival times are zero
    for (int i = 0; i < num; i++) {
        tasks[i].pid = i + 1;
        tasks[i].at = 0;
        cout << "Enter burst time for process " << i + 1 << ": ";
        cin >> tasks[i].bt;
    }
```

```
// Calculate CT, TAT, WT
int time = 0;
for (int i = 0; i < num; i++) {
    if (time < tasks[i].at)
        time = tasks[i].at;
    time += tasks[i].bt;
    tasks[i].ct = time;
    tasks[i].tat = tasks[i].ct - tasks[i].at;
    tasks[i].wt = tasks[i].tat - tasks[i].bt;
}
// Gantt Chart
cout << "\nGantt Chart:\n|";
time = 0;
for (int i = 0; i < num; i++) {
    cout << "  P" << tasks[i].pid << "\t|";
}
cout << "\n0";
for (int i = 0; i < num; i++) {
    time += tasks[i].bt;
    cout << "     \t" << time;
}
cout << "\n";
// Final result table
cout << "\nProcess\tBT\tAT\tCT\tTAT (CT-AT)\tWT (TAT-BT)\n";
double sumTAT = 0, sumWT = 0;
```

```
    for (int i = 0; i < num; i++) {
            cout << "  P" << tasks[i].pid << "\t";
            cout<< tasks[i].bt << "\t";
            cout<< tasks[i].at << "\t";
            cout<< tasks[i].ct << "\t";
            cout<< tasks[i].ct << "-" << tasks[i].at;
            cout<< "=" << tasks[i].tat << "\t\t";
            cout<< tasks[i].tat << "-" << tasks[i].bt;
            cout<< "=" << tasks[i].wt << "\n";
        sumTAT += tasks[i].tat;
        sumWT += tasks[i].wt;
    }
    cout << "\nAverage Turnaround Time = " << sumTAT / num;
    cout << "\nAverage Waiting Time = " << sumWT / num << "\n";
    return 0;
}
```

Output:

```
Enter number of processes: 5
Enter burst time for process 1: 2
Enter burst time for process 2: 4
Enter burst time for process 3: 3
Enter burst time for process 4: 5
Enter burst time for process 5: 6

Gantt Chart:
|  P1  |  P2  |  P3  |  P4  |  P5  |
0      2      6      9      14     20

Process BT     AT     CT     TAT (CT-AT)     WT (TAT-BT)
  P1    2      0      2      2-0=2           2-2=0
  P2    4      0      6      6-0=6           6-4=2
  P3    3      0      9      9-0=9           9-3=6
  P4    5      0      14     14-0=14         14-5=9
  P5    6      0      20     20-0=20         20-6=14

Average Turnaround Time = 10.2
Average Waiting Time = 6.2

c:\Users\Roshan\Desktop\Roshan_OS>
```

Conclusion:

FCFS is simple and fair because every process is executed in the order of arrival. However, it can cause the convoy effect (a long process delays all others). It works well for smaller systems with fewer processes but is not always efficient for modern multitasking operating systems.