MACHINE PROBLEM #1: WRITE A PROGRAM THAT SIMULATES A NON-PREEMPTIVE CPU SCHEDULING ALGORITHMS TO FIND TURN AROUND AND WAITING TIME USING FCFS.

START DATE: AUGUST 13, 2025 **END DATE: AUGUST 25, 2025**

I. OBJECTIVES

Write a program to simulate the non-preemptive CPU scheduling algorithms to find turnaround time and waiting time for the above problem using FCFS.

II.

II. DISCUSSION

CPU Scheduling maybe classified as non-preemptive and preemptive. Non-preemptive scheduling is where the CPU cannot be taken away from its currently executing process. While the preemptive scheduling is when the CPU can be taken away from its currently executing process

Related Terms

1. Turn-Around time - the time between the point a process is submitted and the time it finishes executing
2. Waiting time – the time a process has to spend inside the ready queue waiting to be executed by the CPU
3. FCFS – First come first serve. It is a non-preemptive scheduling algorithm wherein the one that enters the ready queue first gets to be executed by the CPU first.

III. SOFTWARE NEEDED

ITEM NO.	DESCRIPTION	QUANTITY
1	Any programming language	1
2	Web Browser	1

IV. METHODOLOGY

1. Let the user enter the no of process. Valid no of process is from 3-10 only.
2. Enter the process ID, arrival time and the burst time based on the number of process the user wants to compute
3. Then calculate and display the waiting time, turnaround time, average waiting time and average turn-around time of the process accordingly.
4. Provide all possible validations.

V. SAMPLE OUTPUT

Enter the no. of process: 5
Enter process ID for Process1: P1
Enter process ID for Process2: P2
Enter process ID for Process3: P3
Enter process ID for Process4: P4
Enter process ID for Process5: P5
Enter waiting time for P1: 2
Enter waiting time for P2: 0

Enter waiting time for P3: 3
Enter waiting time for P4: 9
Enter waiting time for P5: 6
Enter burst time for P1: 8
Enter burst time for P2: 4
Enter burst time for P3: 2
Enter burst time for P4: 5
Enter burst time for P5: 7

Average waiting: 6.2
Average turn-around time: 11.4

VI. GRADING CRITERIA

Criteria	Weight	Rating
Functionality	45%	
Efficiency	40%	
Code Readability and Standards	15%	
TOTAL	100%	

VII. QUESTIONS

PRE-LAB QUESTIONS

a. Define operating system?

- A system software that manages computer hardware and software resources and provides services for computer programs.
- It creates a virtual machine interface between the user, application program and hardware.
- It acts as the computer's resource manager or resource allocator.
- It functions as the program launcher.

b. What are the different types of operating systems?

- Serial Processing (First Generation)
- Batch Systems (Second Generation)
- Multiprogrammed Systems (Third Generation)
- Time Sharing Systems (Fourth Generation)

/*Alternate answer as I am not sure what operating system are we talking about */

- Computer Operating System (Windows, Linux, etc.)
- Mobile Operating System (Android, Apple, etc.)

c. Define a process?

- A program that is currently running on the computer. It uses the CPU to execute instructions, the memory to store data, and input/output devices to interact with the user or other programs.

d. What is CPU Scheduling?

- Is the process of determining which process from the ready queue will be executed next by the CPU. Its goal is to maximize CPU utilization and minimize waiting time, turnaround time, and response time.

e. Define arrival time, burst time, waiting time, turnaround time?

 Arrival Time: The time when a process enters the ready queue.

 Burst Time: The total time required by a process to execute on the CPU.

 Waiting Time: The total time a process spends waiting in the ready queue before execution.

 Turnaround Time: The total time taken from process arrival to completion.

POST-LAB QUESTIONS

1. How to compute the waiting time and burst time of every process?
   - Waiting Time: Calculated as the difference between the start time of a process and its arrival time. In FCFS: WT = (Start Time−Arrival Time)
   - Burst Time: In FCFS, this is directly given as input; it is the execution time required by the process.
2. How to compute the average waiting time and average burst time of every process?
   - Average Waiting Time: In FCFS, add all the waiting time of the process and divide it based on how many processes the CPU executed.
   - Average Burst Time: In FCFS, add all the burst time of the process and divide it based on how many processes the CPU executed.

VIII. CONCLUSION

In this laboratory exercise, we simulated non-preemptive CPU scheduling using the First Come, First Serve (FCFS) algorithm. The experiment allowed us to calculate waiting time and turnaround time for each process and then determine their averages. We learned that in non-preemptive scheduling, once a process is assigned to the CPU, it cannot be interrupted until completion.

The FCFS algorithm is simple to implement but may lead to longer waiting times if processes with long burst times arrive first.This activity helped us understand how scheduling affects CPU efficiency and how turnaround time and waiting time are computed, reinforcing the importance of process scheduling in operating systems.

C++ Source Code:

```
#include <iostream>
#include <iomanip>
#include <vector>
#include <set>
#include <string>
#include <algorithm>
using namespace std;

class Process
{
```

```
private:
    string processID;
    int arrivalTime;
    int burstTime;
    int waitingTime;
    int turnaroundTime;

public:
    Process(string id = "", int at = 0, int bt = 0)
    {
        processID = id;
        arrivalTime = at;
        burstTime = bt;
        waitingTime = 0;
        turnaroundTime = 0;
    }

    // Getters
    string getProcessID() const { return processID; }
    int getArrivalTime() const { return arrivalTime; }
    int getBurstTime() const { return burstTime; }
    int getWaitingTime() const { return waitingTime; }
    int getTurnaroundTime() const { return turnaroundTime; }

    // Setters
    void setWaitingTime(int wt) { waitingTime = wt; }
    void setTurnaroundTime(int tat) { turnaroundTime = tat; }
};

// Function to compute waiting time and turnaround time (FCFS)
void computeTimes(vector<Process> &processes)
{
    int n = processes.size();
    int currentTime = 0;

    for (int i = 0; i < n; i++)
    {
        if (currentTime < processes[i].getArrivalTime())
        {
            currentTime = processes[i].getArrivalTime();
        }

        int wt = currentTime - processes[i].getArrivalTime();
        if (wt < 0)
            wt = 0;
        processes[i].setWaitingTime(wt);

        currentTime += processes[i].getBurstTime();
        int tat = processes[i].getBurstTime() + processes[i].getWaitingTime();
        processes[i].setTurnaroundTime(tat);
    }
}

// Function to print results in a table
void createTable(const vector<Process> &processes)
{
    cout << "--------------------------------------------------------------------\n";
    cout << "| Process ID   | Arrival Time | Burst Time | Waiting Time | Turnaround Time |\n";
    cout << "--------------------------------------------------------------------\n";

    double totalWT = 0, totalTAT = 0;
```

```
    for (const auto &p : processes)
    {
        cout << setw(10) << p.getProcessID()
             << setw(14) << p.getArrivalTime()
             << setw(14) << p.getBurstTime()
             << setw(14) << p.getWaitingTime()
             << setw(16) << p.getTurnaroundTime() << "\n";

        totalWT += p.getWaitingTime();
        totalTAT += p.getTurnaroundTime();
    }

    cout << "-------------------------------------------------------------------------\n";
    cout << "Average Waiting Time     : " << fixed << setprecision(2) << (totalWT / processes.size())
<< "\n";
    cout << "Average Turnaround Time  : " << fixed << setprecision(2) << (totalTAT /
processes.size()) << "\n";
}

int main()
{
    int processCount;

    // Validate number of processes
    do
    {
        cout << "Enter number of processes (3-10): ";
        cin >> processCount;
        if (cin.fail() || processCount < 3 || processCount > 10)
        {
            cout << "Invalid input! Please enter a number between 3 and 10.\n";
            cin.clear();
            cin.ignore(1000, '\n');
        }
    } while (processCount < 3 || processCount > 10);

    vector<Process> processes;
    set<string> usedIDs;

    for (int i = 0; i < processCount; i++)
    {
        string id;
        int at, bt;

        cout << "\nEnter details for Process " << (i + 1) << ":\n";

        // Process ID validation
        while (true)
        {
            cout << "Process ID: ";
            cin >> id;

            transform(id.begin(), id.end(), id.begin(), ::toupper);

            if (usedIDs.count(id))
            {
                cout << "Process ID already exists! Enter a unique Process ID.\n";
                continue;
            }
            usedIDs.insert(id);
            break;
```

```
        }

        // Arrival Time validation
        cout << "Arrival Time: ";
        while (!(cin >> at) || at < 0)
        {
            cout << "Invalid input! Arrival Time must be >= 0: ";
            cin.clear();
            cin.ignore(1000, '\n');
        }

        // Burst Time validation
        cout << "Burst Time: ";
        while (!(cin >> bt) || bt <= 0)
        {
            cout << "Invalid input! Burst Time must be > 0: ";
            cin.clear();
            cin.ignore(1000, '\n');
        }

        processes.emplace_back(id, at, bt);
    }

    // Compute times using FCFS
    computeTimes(processes);

    // Print results
    createTable(processes);

    return 0;
}
```