Experiment Number: 04

Experiment Name: Implement the First-Come-First-Serve (FCFS) Scheduling Algorithm.

Objectives

- To understand the basic working mechanism of the First-Come-First-Serve (FCFS) scheduling algorithm.
- To calculate Completion Time, Turnaround Time, and Waiting Time for a set of processes.
- To implement FCFS scheduling in C programming and analyze the output for different process inputs.
- To learn how process scheduling can affect system performance in terms of waiting and turnaround times.

Theory

The **First-Come-First-Serve (FCFS)** scheduling algorithm is one of the simplest CPU scheduling algorithms. In this method:

- The process that arrives first is executed first.
- If two processes arrive at the same time, they are executed in the order they appear in the queue.
- Once a process starts executing, it runs until completion without preemption.

Key Terms:

1. **Arrival Time**: The time at which a process enters the system and is ready for execution.
2. **Burst Time**: The total time required by a process for execution on the CPU.
3. **Completion Time**: The time at which a process finishes execution.
4. **Turnaround Time (TAT)**: The total time taken from a process's arrival to its completion.

$$\text{Turnaround Time} = \text{Completion Time} - \text{Arrival Time}$$

5. **Waiting Time (WT)**: The amount of time a process spends waiting in the ready queue before getting CPU time.

$$\text{Waiting Time} = \text{Turnaround Time} - \text{Burst Time}$$

Since FCFS is non-preemptive, once a process starts, it will run until completion before another process can start. FCFS can lead to the **convoy effect**, where short processes get stuck waiting for longer processes to complete, increasing the average waiting time.

Algorithm Steps:

1. **Input the Processes**:
    - Take input for the number of processes, n.
    - For each process, input:
        - **Arrival Time**: Time at which the process arrives.
        - **Burst Time**: Time required by the process to complete its execution.
2. **Sort Processes by Arrival Time**:
    - Arrange the processes in ascending order of their **Arrival Times**. If two processes have the same arrival time, they are scheduled based on the order they appear in the input.
3. **Initialize Completion Time of the First Process**:
    - Set the **Completion Time** of the first process as Arrival Time + Burst Time since it starts executing as soon as it arrives.
4. **Calculate Completion Time for Each Process**:
    - For each subsequent process i:
        - If the **Arrival Time** of process i is greater than the **Completion Time** of the previous process (i - 1), then the process can start executing as soon as it arrives: Completion Time[i]=Arrival Time[i]+Burst Time[i]\text{Completion Time}[i] = \text{Arrival Time}[i] + \text{Burst Time}[i]Completion Time[i]=Arrival Time[i]+Burst Time[i]
        - Otherwise, if the process arrives while the previous process is still executing, it will have to wait until the previous process completes. Hence: Completion Time[i]=Completion Time[i−1]+Burst Time[i]\
5. **Calculate Turnaround Time for Each Process**:

- For each process, **Turnaround Time** (TAT) is the total time from its arrival until its completion:
- Turnaround Time[i]=Completion Time[i]−Arrival Time[i]

6. **Calculate Waiting Time for Each Process**:
   - For each process, **Waiting Time** (WT) is the time spent in the ready queue before getting CPU time, calculated as:
   - Waiting Time[i]=Turnaround Time[i]−Burst Time[i
7. **Calculate Averages**:
   - Compute the **Average Waiting Time** by summing up the waiting times of all processes and dividing by the number of processes.
   - Compute the **Average Turnaround Time** similarly.
8. **Output**:
   - Display the **Completion Time**, **Turnaround Time**, and **Waiting Time** for each process.
   - Display the **Average Waiting Time** and **Average Turnaround Time**.

Source Code:

```
#include <stdio.h>


int main() {
    int n, i;
    float avg_waiting_time = 0, avg_turnaround_time = 0;


    printf("Enter the number of processes: ");
    scanf("%d", &n);


    int arrival_time[n], burst_time[n], completion_time[n], waiting_time[n], turnaround_time[n];
```

```
    // Input burst time and arrival time for each process
    printf("Enter Arrival Time and Burst Time for each process:\n");
    for(i = 0; i < n; i++) {
        printf("Process %d:\n", i + 1);
        printf("Arrival Time: ");
        scanf("%d", &arrival_time[i]);
        printf("Burst Time: ");
        scanf("%d", &burst_time[i]);
    }

    // Calculate completion time for each process
    completion_time[0] = arrival_time[0] + burst_time[0]; // First process
    for(i = 1; i < n; i++) {
        if (arrival_time[i] > completion_time[i - 1]) {
            completion_time[i] = arrival_time[i] + burst_time[i]; // Process arrives after
previous completion
        } else {
            completion_time[i] = completion_time[i - 1] + burst_time[i]; // Process can
start after previous completion
        }
    }

    // Calculate turnaround time and waiting time for each process
    for(i = 0; i < n; i++) {
        turnaround_time[i] = completion_time[i] - arrival_time[i];
        waiting_time[i] = turnaround_time[i] - burst_time[i];
        avg_waiting_time += waiting_time[i];
```

```
        avg_turnaround_time += turnaround_time[i];
    }

    // Calculate average times
    avg_waiting_time /= n;
    avg_turnaround_time /= n;

    // Display the results
    printf("\nProcess\tArrival Time\tBurst Time\tCompletion Time\tTurnaround Time\tWaiting Time\n");
    for(i = 0; i < n; i++) {
        printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", i + 1, arrival_time[i], burst_time[i], completion_time[i], turnaround_time[i], waiting_time[i]);
    }

    printf("\nAverage Waiting Time: %.2f", avg_waiting_time);
    printf("\nAverage Turnaround Time: %.2f\n", avg_turnaround_time);

    return 0;
}
```

Input and output:

```
"C:\Users\MKA_SAGAR\Desktop\Summer-2024\OS_Lab Manual\MKA\Algorithm_LAB\FCFS.exe"
Enter the number of processes: 5
Enter Arrival Time and Burst Time for each process:
Process 1:
Arrival Time: 0
Burst Time: 2
Process 2:
Arrival Time: 1
Burst Time: 6
Process 3:
Arrival Time: 2
Burst Time: 4
Process 4:
Arrival Time: 3
Burst Time: 9
Process 5:
Arrival Time: 6
Burst Time: 12
```

Process	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time
1	0	2	2	2	0
2	1	6	8	7	1
3	2	4	12	10	6
4	3	9	21	18	9
5	6	12	33	27	15

```
Average Waiting Time: 6.20
Average Turnaround Time: 12.80
```