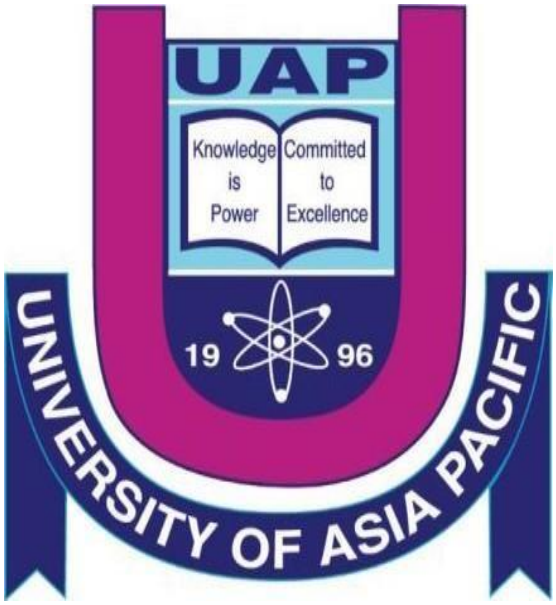

Department of Computer Science & Engineering

Course Title: Operating System Lab

Course Code: CSE 406

Lab Report No: 01

Lab Report: First-Come, First-Served (FCFS) Scheduling

Submission Date: 19-02-2025

Submitted To:
Atia Rahman Orthi
Lecturer,
Department of CSE, UAP

Submitted By:
Name: Amirul Islam Papon
Reg No: 21201076
Sec: B

Problem Statement: The objective of this lab is to implement and analyze the First-Come, First-Served (FCFS) scheduling algorithm. FCFS is a non-preemptive scheduling technique where processes are executed in the order they arrive. The goal is to calculate the Completion Time (CT), Turnaround Time (TAT), and Waiting Time (WT) for a set of processes and evaluate the efficiency of the scheduling method.

Steps to Implement FCFS Scheduling:

1. **Input Process Details**
    - Read the number of processes.
    - Input the Process ID, Arrival Time (AT), and Burst Time (BT) for each process.
2. **Sort Processes by Arrival Time**
    - Arrange processes in ascending order of AT to ensure they execute in the correct sequence.
3. **Compute Completion Time (CT)**
    - The first process starts execution at its arrival time.
    - Each subsequent process starts after the previous one completes.
4. **Calculate Turnaround Time (TAT)**
    - Turnaround Time is the total time a process spends in the system.
    - Formula: TAT = CT – AT
5. **Calculate Waiting Time (WT)**
    - Waiting Time is the time a process spends waiting in the ready queue before execution.
    - Formula: TAT – BT

Source Code:

```python
def fcfs_scheduling(processes):
    processes.sort(key=lambda x: x[1])

    n = len(processes)
    tat = [0] * n
    wt = [0] * n
    ct = [0] * n

    ct[0] = processes[0][1] + processes[0][2]
    for i in range(1, n):
        ct[i] = max(ct[i-1], processes[i][1]) + processes[i][2]

    for i in range(n):
        tat[i] = ct[i] - processes[i][1]
        wt[i] = tat[i] - processes[i][2]

    print("Process\tTAT\tWT")
    for i in range(n):
        print(f"{processes[i][0]}\t\t{tat[i]}\t{wt[i]}")


process_list = [
    ('P1', 2, 5),
    ('P2', 0, 3),
    ('P3', 4, 4),
]

fcfs_scheduling(process_list)
```

Output:

```
[Running] python -u "d:\Operating System\fcfs_scheduling2.py"
Process TAT WT
P2      3   0
P1      6   1
P3      8   4

[Done] exited with code=0 in 0.07 seconds
```

Discussion: FCFS scheduling is simple and easy to implement but suffers from the convoy effect, where shorter processes wait for longer processes to complete. This results in higher waiting times and low CPU efficiency in certain scenarios. Despite its limitations, FCFS is useful for batch processing systems.

The experiment demonstrates that FCFS can cause high average waiting time if shorter processes arrive after longer ones.

Class Work Source Code: https://github.com/Amirul-Islam-Papon/Operating-Syatem/blob/main/fcps_scheduling.py
Assignment Source Code: https://github.com/Amirul-Islam-Papon/Operating-Syatem/blob/main/fcfs_scheduling2.py