School of Engineering and Technology
Operating Systems

Laboratory Exercise No.1
Fundamentals of Operating System

JACOB S. SANTOS **09/17/24**

BSCOE 4-1 **Engr. Elizier Obamos**

Objectives: At the end of the activity, the students must be able to understand and implement the First-Come, First-Served (FCFS) CPU scheduling algorithm. Students will calculate average waiting time and turnaround time for a set of processes using the FCFS scheduling technique.

Materials Needed:

- Computer with Python (or any programming language) installed.
- Text editor or IDE (e.g., VSCode, PyCharm, etc.).
- Pen and paper for manual calculations (optional).

Concept

In the First-Come, First-Served (FCFS) scheduling algorithm, the process that arrives first gets the CPU first. It is a non-preemptive scheduling algorithm, meaning that once a process starts executing, it runs to completion without being interrupted. The simplicity of the algorithm makes it easy to implement, but it can lead to inefficiencies like the convoy effect.

Key Terms

- Burst **Time (BT):** The time required by a process to complete its execution.
- Waiting **Time (WT)**: The time a process spends in the ready queue before getting executed.
- Turnaround **Time (TAT):** The total time taken by a process from arrival to completion. TAT =WT + BT.

Part 1: Manual Calculation

1. Consider the following processes with their respective arrival time and burst times:

Process ID	Arrival Time	Burst Time
A	0	5
B	2	3
C	4	8
D	3	6

2. Using FCFS scheduling, determine:
- The waiting time for each process.
- The turnaround time for each process.
- The average waiting time and turnaround time.

3. Write down the order of execution and manually compute the waiting and turnaround times for each process based on the arrival order.

Part 2: Implementing FCFS with Arrival Time in Python

Task 1: Write a program to simulate the FCFS scheduling algorithm with arrival times.

Task 2: Modify the program to accept user input for the number of processes (Limit to 3-6), their arrival times, and burst times.

o Modify the arrival time and burst time lists to accept user input.
o Add input validation (e.g., ensuring that arrival times and burst times are non-negative).
o Re-run the program with different inputs.

Task 3: Analyze the Results
- Run the program with different sets of arrival and burst times.
- Compare the results of waiting time and turnaround time for different inputs.
- Discuss how arrival times affect the scheduling and how FCFS handles processes arriving at different times.

Task4: Screenshot the Output:

```
Enter the number of processes (3-6): 5
Enter arrival times and burst times:
Process 1 Arrival Time: 0
Process 1 Burst Time: 8
Process 2 Arrival Time: 2
Process 2 Burst Time: 4
Process 3 Arrival Time: 2
Process 3 Burst Time: 9
Process 4 Arrival Time: 4
Process 4 Burst Time: 5
Process 5 Arrival Time: 6
Process 5 Burst Time: 3
Process ID      Arrival Time    Burst Time      Waiting Time    Turnaround Time
A               0               8               0               8
B               2               4               6               10
C               2               9               10              19
D               4               5               17              22
E               6               3               20              23

Gantt Chart:
0        8       12      21      26      29
A       B       C       D       E

Average Waiting Time: 10.60
Average Turnaround Time: 16.40
```

Discussion Questions:

1. How does the arrival time affect the waiting time for each process?

Since FCFS executes processes in the order they arrive, a process that arrives earlier gets executed sooner, while later-arriving processes must wait until all previously arrived processes finish. This means that if a process arrives after several others, it will experience a longer waiting time, even if it requires less execution time. This can lead to inefficiencies especially in cases where long processes block shorter ones from executing quickly.

2. What happens when a process arrives after the previous process has already completed? How does the FCFS algorithm handle this scenario?

The CPU will be idle until the new process arrives. Since FCFS schedules processes in the order of their arrival, it will immediately start executing the new arrived process as soon as it becomes available. This scenario highlights one of the limitations of FCFS, which is idleness which there is gaps between process arrivals, as the algorithm does not anticipate or optimize for such delays. It waits for the next process to arrive and then executes it.

3. How does FCFS compare to other scheduling algorithms, such as Shortest Job First (SJF) or Round Robin, when arrival times vary?

FCFS is simple and can lead to longer waiting times for shorter processes. SJF reduces average waiting time but can be unfair to longer tasks, and Round Robn offers a more balanced time-shared approach, though it comes with its own trade off with efficiency.