📅 Week 7 – CSIS 2260 (Operating Systems) – Survey Week with Practice

Day 4 – Processor Management & Scheduling

🔍 Concepts

- **CPU Scheduling Goals**:
  - Maximize CPU utilization
  - Minimize waiting time
  - Provide fairness among processes
- **Common Scheduling Algorithms**:
  - **First-Come, First-Served (FCFS)** – Non-preemptive, simple queue.
  - **Shortest Job First (SJF)** – Runs process with the smallest CPU burst next.
  - **Round Robin (RR)** – Preemptive, each process gets a fixed time quantum.
- **Context Switching**:
  - Saving the state of the current process and loading the state of the next process.

📖 Theory Notes

- **FCFS**:
  - Pros: Simple
  - Cons: Can cause "convoy effect"
- **SJF**:
  - Pros: Minimizes average waiting time
  - Cons: Requires knowing the CPU burst time in advance

- **RR**:
  - Pros: Fair time allocation
  - Cons: Too small quantum → overhead; Too large quantum → behaves like FCFS

✍️ Pseudocode Test (Round Robin)

```
WHILE processes remain
  SELECT next process in queue
  EXECUTE for time quantum
  IF process finished THEN remove
  ELSE add to end of queue
ENDWHILE
```

🧪 Code Simulation (Python Example)

```python
from collections import deque

processes = deque([
    {"name": "P1", "time": 5},
    {"name": "P2", "time": 3},
    {"name": "P3", "time": 8}
])

time_quantum = 2

while processes:
    process = processes.popleft()
    print(f"Running {process['name']} for {min(time_quantum,
        process['time'])} units")
    process['time'] -= time_quantum
```

```
    if process['time'] > 0:
        processes.append(process)
```

🧪 Code Simulation (JavaScript Example)

```javascript
let processes = [
  { name: "P1", time: 5 },
  { name: "P2", time: 3 },
  { name: "P3", time: 8 }
];
const timeQuantum = 2;

while (processes.length > 0) {
  let process = processes.shift();
  console.log(`Running ${process.name} for
        ${Math.min(timeQuantum, process.time)} units`);
  process.time -= timeQuantum;
  if (process.time > 0) {
    processes.push(process);
  }
}
```

🧪 Code Simulation (C# Example)

```csharp
using System;
using System.Collections.Generic;

class Program
{
    static void Main()
    {
```

```
        var processes = new Queue<(string Name, int Time)>
        (new[]
        {
            ("P1", 5),
            ("P2", 3),
            ("P3", 8)
        });

        int timeQuantum = 2;

        while (processes.Count > 0)
        {
            var process = processes.Dequeue();
            int runTime = Math.Min(timeQuantum,
        process.Time);
            Console.WriteLine($"Running {process.Name} for
        {runTime} units");
            int remaining = process.Time - timeQuantum;
            if (remaining > 0)
            {
                processes.Enqueue((process.Name, remaining));
            }
        }
    }
}
```

❓ Quiz

1. Which algorithm can cause starvation and why?
2. In Round Robin, what is the impact of choosing a very small time quantum?
3. What is the “convoy effect” in FCFS scheduling?

✅ Deliverable for Day 4

- Notes on FCFS, SJF, and RR algorithms.
- Completed pseudocode test.
- Working code simulations in Python, JavaScript, and C#.
- Quiz answers recorded.