**Report**

**Multilevel Queue Scheduling Algorithm in C++**

**23000066**

---

## Introduction

This report is a reference for the program I created to simulate a **multilevel queue scheduling algorithm**.

Here I'll discuss the program's logic, sample inputs and outputs, analyze the results, and the strengths and weaknesses of each scheduling algorithm used.

---

## LOGIC

The program simulates a CPU scheduler that manages processes using **4 queues**, each with a different priority and scheduling algorithm:

1. **Queue 0 (q0)**: Highest priority, uses **Round Robin (RR)** with a time quantum of 20 seconds.
2. **Queue 1 (q1)**: Medium-high priority, uses **Shortest Job First (SJF)**.
3. **Queue 2 (q2)**: Medium-low priority, uses **Shortest Job First (SJF)**.
4. **Queue 3 (q3)**: Lowest priority, uses **First-In-First-Out (FIFO)**.

The CPU switches between queues every 20 seconds, ensuring higher-priority processes are executed first while still giving lower-priority processes a chance to run.

---

**Sample Test Cases**

**Input:**

```
PS C:\Users\user\Documents\UCSC\Academics\OS\week8_MultiLevelQueue\23000066>
 g++ 23000066_MLQ.cpp
PS C:\Users\user\Documents\UCSC\Academics\OS\week8_MultiLevelQueue\23000066>
 .\a.exe
Enter the number of processes: 4
Enter details for process 1:
Burst time: 10
Arrival time: 0
Priority (0-3): 0
Enter details for process 2:
Burst time: 20
Arrival time: 0
Priority (0-3): 0
Enter details for process 3:
Burst time: 30
Arrival time: 0
Priority (0-3): 0
Enter details for process 4:
Burst time: 40
Arrival time: 0
Priority (0-3): 0
```

**Output:**

```
Process ID      Waiting Time    Turnaround Time
1               0               10
2               60              70
3               70              80
4               80              100
PS C:\Users\user\Documents\UCSC\Academics\OS\week8_MultiLevelQueue\23000066>
```

**Analysis:**

1. **Process 1** (q0 - Round Robin):
   - Runs immediately and completes in 10 seconds.
   - Waiting Time: 0 (since it starts right away).
   - Turnaround Time: 10 (completion time - arrival time).
2. **Process 2** (q1 - SJF):
   - Runs after q0 is empty.
   - Completes at time 80.
   - Waiting Time: 60 (time spent waiting for q0 to finish).
   - Turnaround Time: 70.

3. **Process 3** (q2 - SJF):
   - Runs after q1 is empty.
   - Completes at time 160.
   - Waiting Time: 70.
   - Turnaround Time: 80.
4. **Process 4** (q3 - FIFO):
   - Runs last.
   - Completes at time :260.
   - Waiting Time: 80.
   - Turnaround Time: 100.

---

---

**Analysis of Scheduling Algorithms**

1. **Round Robin (RR)**:
   - **Pros**: Fair to all processes in the queue. No process is left waiting indefinitely.
   - **Cons**: High overhead due to frequent context switching.
   - **Best For**: Time-sharing systems where fairness is important.
2. **Shortest Job First (SJF)**:
   - **Pros**: Minimizes average waiting time by running shorter jobs first.
   - **Cons**: Can cause starvation for longer processes if shorter processes keep arriving.
   - **Best For**: Systems where shorter tasks need to be completed quickly.
3. **First-In-First-Out (FIFO)**:
   - **Pros**: Simple to implement. No complex logic required.
   - **Cons**: Longer processes can delay shorter ones, leading to higher waiting times.
   - **Best For**: Systems where simplicity is more important than efficiency.

---

**Conclusion**

The multilevel queue scheduling algorithm effectively balances **priority** and **fairness**:

- Higher-priority processes (e.g., q0) are executed first.
- Lower-priority processes (e.g., q3) still get a chance to run, but only after higher-priority queues are empty.

**Strengths**:

- Ensures critical processes (high priority) are handled quickly.
- Provides a balance between fairness and efficiency.

**Limitations**:

- Lower-priority processes may experience **starvation** if higher-priority queues are always busy.
- The fixed time quantum (20 seconds) may not be optimal for all workloads. Program crashes and occur memory leaks in some times.

---

**Final Thoughts**

This program demonstrates how multilevel queue scheduling can be used to manage processes with varying priorities. By combining different scheduling algorithms, it achieves a good balance between responsiveness and fairness. However, in real-world systems, dynamic adjustments (e.g., varying time quanta or priority boosting) may be needed to further improve performance.