Programming Project 3 Report

Author: Oscar Lopez
Class: CS 433 Operating Systems | Section 1
Professor: Dr. Zhang

Summary

This report documents the implementation of various CPU scheduling algorithms as part of Assignment 3. The project implements five scheduling algorithms: First-Come-First-Served (FCFS), Shortest-Job-First (SJF), Priority, Round-Robin (RR), and Priority with Round-Robin (Priority-RR). The implementation follows object-oriented design principles with a base Scheduler class and algorithm-specific subclasses. All algorithms have been tested with the provided test cases and produce the expected outputs.

Implementation Details

Architecture Overview

The implementation follows an object-oriented approach with a class hierarchy:

1. **PCB Class**: Represents a process with attributes such as ID, name, priority, burst time, and arrival time.
2. **Scheduler Base Class**: An abstract class defining the interface for all scheduling algorithms.
3. **Algorithm-Specific Subclasses**: Concrete implementations for each scheduling algorithm.

Class Structure

PCB Class

- Stores process information including ID, name, priority, burst time, and arrival time
- Provides accessors and mutators for these attributes
- Used to represent processes in the ready queue

Scheduler Base Class

- Defines pure virtual functions: `init()`, `simulate()`, and `print_results()`
- Maintains common statistics like total waiting time and turnaround time
- Provides a consistent interface for all scheduling algorithms

Algorithm-Specific Subclasses

1. **SchedulerFCFS**:
    - Implements First-Come-First-Served scheduling
    - Processes are executed in the order they arrive
    - Uses a standard queue data structure
2. **SchedulerSJF**:
    - Implements Shortest-Job-First scheduling
    - Processes are executed in order of burst time
    - Uses a priority queue based on burst time
3. **SchedulerPriority**:
    - Implements Priority scheduling
    - Processes are executed in order of priority (higher number = higher priority)
    - Uses a priority queue based on process priority
4. **SchedulerRR**:
    - Implements Round-Robin scheduling
    - Processes are executed for a time quantum or until completion
    - Uses a queue with cycling behavior
5. **SchedulerPriorityRR**:
    - Implements Priority with Round-Robin scheduling
    - Processes with the same priority are scheduled using Round-Robin
    - Uses a multi-level queue structure

Key Implementation Features

Synchronization and State Management

- Each scheduler maintains its own ready queue and process list
- The `simulate()` function handles the core scheduling logic
- Process state transitions are tracked throughout execution

Statistics Collection

- Waiting time and turnaround time are calculated for each process
- Average waiting time and turnaround time are computed across all processes
- Results are displayed in a consistent format

Time Management

- A global clock is maintained to track execution time
- Time slices are managed for preemptive algorithms (RR and Priority-RR)
- Process execution is simulated with appropriate time accounting

Features

Core Features

1. **Multiple Scheduling Algorithms**: Implements five different scheduling algorithms
2. **Consistent Interface**: All algorithms follow the same interface pattern
3. **Detailed Statistics**: Provides per-process and average timing statistics
4. **Configurable Parameters**: Time quantum for RR can be specified via command line
5. **Process Visualization**: Shows the execution sequence of processes

Advanced Features

1. **Priority-Based Round-Robin**: Combines priority scheduling with round-robin for processes of equal priority
2. **Extensible Design**: New scheduling algorithms can be added by extending the base class
3. **Robust Error Handling**: Validates input parameters and handles edge cases
4. **Memory Management**: Proper cleanup of resources in destructors

Performance Analysis

Time Complexity

- FCFS: O(n) where n is the number of processes
- SJF: O(n log n) due to priority queue operations
- Priority: O(n log n) due to priority queue operations
- RR: O(n * total_burst_time / time_quantum)
- Priority-RR: O(n log n + n * total_burst_time / time_quantum)

Space Complexity

- All algorithms: O(n) where n is the number of processes

Comparative Analysis

Based on the test cases, the algorithms perform as expected: - FCFS has the simplest implementation but often results in higher average waiting times - SJF provides optimal average waiting time for the given workload - Priority scheduling favors high-priority processes regardless of burst time - RR provides fair CPU distribution but increases context switching overhead - Priority-RR balances priority requirements with fairness

Limitations and Drawbacks

1. **Arrival Time Assumption**: The implementation assumes all processes arrive at time 0

2. **No I/O Consideration**: The simulation does not account for I/O operations or blocking
3. **Static Burst Time**: Burst times are fixed and known in advance (not realistic in practice)
4. **No Process Dependencies**: The implementation does not handle process dependencies or deadlocks
5. **Limited Preemption**: SJF and Priority implementations are non-preemptive
6. **No Dynamic Priority**: Process priorities remain static throughout execution

Usage Instructions

Compilation

The project includes a Makefile with targets for each scheduling algorithm:

```
make fcfs    # Compile FCFS scheduler
make sjf     # Compile SJF scheduler
make priority # Compile Priority scheduler
make rr      # Compile RR scheduler
make priority_rr # Compile Priority-RR scheduler
```

Execution

Each executable takes a file containing process information as input:

```
./fcfs schedule.txt
./sjf schedule.txt
./priority schedule.txt
./rr schedule.txt 6  # 6 is the time quantum
./priority_rr schedule.txt 6  # 6 is the time quantum
```

Input File Format

The input file should contain one process per line with the format:

```
<process_name> <priority> <burst_time>
```

Example:

```
T1 4 20
T2 3 25
T3 3 25
T4 5 15
T5 5 20
T6 1 10
```

```
T7 3 30
T8 10 25
```

Output Format

The program outputs: 1. Process information (name, priority, burst time) 2. Execution sequence 3. Per-process turnaround and waiting times 4. Average turnaround and waiting times

Conclusion

The implementation successfully meets all the requirements for Assignment 3. It provides a comprehensive set of CPU scheduling algorithms with detailed statistics and visualization. The object-oriented design ensures code reusability and extensibility. While there are some limitations inherent to the simulation approach, the implementation serves as an effective educational tool for understanding CPU scheduling algorithms.