Assignment 2 – Process Scheduling and Memory Management

Instructor: Gabriel Wainer

Course: SYSC 4001 – Operating Systems

Student 1: Grant Phillips, 101230563

Student 2: Braeden Kloke, 100895984

­

Submitted: XXX, 2023

Due: Nov 13, 2023

Table of Contents

[Part 1 Concepts 3](#_Toc150193626)

[P1.a 3](#_Toc150193627)

[P1.b 3](#_Toc150193628)

[P1.b.i – FCFS 3](#_Toc150193629)

[P1.b.ii – Round Robin 4](#_Toc150193630)

[P1.b.iii – Multiple Queues with Feedback 4](#_Toc150193631)

[P1.d 4](#_Toc150193632)

[P1.e 4](#_Toc150193633)

[P1.e.a - FCFS 4](#_Toc150193634)

[P1.e.b - RR 4](#_Toc150193635)

[P1.e.c – Multi-level Feedback Queues 4](#_Toc150193636)

[P1.f 5](#_Toc150193637)

[Part 2 5](#_Toc150193638)

[P2.iii – Simulation Execution 5](#_Toc150193639)

[P2.iv – Memory Management Simulation Analysis 5](#_Toc150193640)

[Objective 5](#_Toc150193641)

[Experiment Design 5](#_Toc150193642)

[Results 6](#_Toc150193643)

[Conclusion 6](#_Toc150193644)

[Appendix A: Gantt Charts 7](#_Toc150193645)

# Part 1 Concepts

## P1.a

**What are the possible events that can make that process abandon the use of the CPU? Explain how the OS Kernel will react to these different events in detail. Consider the kind of scheduler the system is using.**

Event 1: Interrupt

When an interrupt is detected and the interrupt has priority, then the running process is moved to the READY state while the interrupt is serviced.

Since the CPU scheduler is using a Round Robin with Priorities algorithm, then this interrupt could be the timer interrupting the process when the set quantum is reached.

Then the CPU scheduler picks one process from the READY state, based on highest priority user, to be dispatched to the CPU.

Event 2: I/O Request

When a process makes an I/O request, it is moved to the WAITING state. Then the CPU scheduler picks one process from the READY state, based on highest priority user, to be dispatched to the CPU.

Event 3: Process completes.

When the process finishes executing it moves to the terminated state. This is done by invoking an exit system call. Then the CPU scheduler picks one process from the READY state, based on highest priority user, to be dispatched to the CPU.

## P1.b

We are assuming that the Gantt charts are not being marked. However, the Gantt charts have been included in the Appendix A.

The following were computed using the formula:

### P1.b.i – FCFS

Mean turnaround time = 29.6 seconds

### P1.b.ii – Round Robin

Mean turnaround time = 41.6 seconds

### P1.b.iii – Multiple Queues with Feedback

Mean turnaround time = 40.2 seconds

## P1.d

Coming soon …

## P1.e

**Explain, in general, the differences in the degree to which the following scheduling algorithms discriminate in favour of short processes.**

All algorithms are analyzed on a degree of 0 to 1, where 0 is not discriminatory in favour of short processes and 1 is very discriminatory in favour of short processes.

### P1.e.a - FCFS

FCFS does not discriminate in favour of short processes, thus we have assigned it a degree of 0.

### P1.e.b - RR

RR slightly discriminates in favour of short processes, thus we have assigned it a degree of 0.5.

When a process with CPU time that is longer than the quantum, the process will be kicked out of the CPU and be put at the back of the ready queue. This means long processes will need to re-enter the queue multiple times to run to completion, while short processes with a CPU time less than the quantum will run to completion when allocated to the CPU. We consider this favouring short processes. However, the algorithm is still fair because long processes will eventually run. Thus, we say it slightly discriminates in favour of short processes.

### P1.e.c – Multi-level Feedback Queues

MLFQ discriminates in favour of short processes, thus we have assigned it a degree of 1.

MLFQ is not fair because long processes will end up in the low priority queue and processes in the low priority queue may never run.

This algorithm heavily favours short processes (processes with a CPU run time less than the quantum of the highest priority queue) because they will always be assigned to the highest priority queue and always run to completion. Thus, we say this algorithm is very discriminatory in favour of short processes.

## P1.f

Coming soon …

# Part 2

## P2.iii – Simulation Execution

Coming soon …

## P2.iv – Memory Management Simulation Analysis

### Objective

Analyze the memory schemas in Figure M1 and make a conclusion about which schema has a lower mean time for processes waiting in NEW state.

|  |  |
| --- | --- |
| **Memory Schema 1** | **Memory Schema 2** |
| 500 Mb | 300 Mb |
| 250 Mb | 300 Mb |
| 150 Mb | 350 Mb |
| 100 Mb | 50 Mb |

Figure M1: Memory Schemas

### Experiment Design

10 scenarios will be simulated using a FCFS scheduling algorithm. Each scenario will be run against each memory schema to see which one has a lower mean time for processes waiting in the NEW state.

Each scenario will contain 10 processes. All processes will have identical attributes except for (1) PID and (2) Process Size. All processes will arrive at 0 ms to create a backlog in the NEW state, perform no I/O, and have a total CPU time of 1 ms. The processes will be sorted by process size from lowest to highest.

Figure M2 below contains a summary of each scenario. All scenarios can be viewed in the simulator project folder in **/resources/memory-simulation/**

|  |  |
| --- | --- |
| **Scenario #** | **Description** |
| 0 | 100% of processes at size = 350 Mb. |
| 1 | 100% of processes at size = 300 Mb. |
| 2 | 100% of processes at size = 250 Mb. |
| 3 | 100% of processes at size = 150 Mb. |
| 4 | 100% of processes at size = 100 Mb. |
| 5 | 30% of processes at size = 350 Mb. Remaining processes at size = 50 Mb. |
| 6 | 30% of processes at size = 300 Mb. Remaining processes at size = 50 Mb. |
| 7 | 30% of processes at size = 250 Mb. Remaining processes at size = 50 Mb. |
| 8 | 30% of processes at size = 150 Mb. Remaining processes at size = 50 Mb. |
| 9 | 30% of processes at size = 100 Mb. Remaining processes at size = 50 Mb. |

Figure M2: Scenarios

### Results

Figure M3 below shows the calculated mean time spent in the NEW state for each scenario. Figure M4 below shows the average of the results in M3.

|  |  |  |
| --- | --- | --- |
| Scenario | Mean Time Spent in NEW State Using Memory Schema 1 (ms) | Mean Time Spent in NEW State Using Memory Schema 2 (ms) |
| 0 | 4.5 | 4.5 |
| 1 | 4.5 | 2.8 |
| 2 | 3.6 | 2.8 |
| 3 | 2.8 | 2.8 |
| 4 | 2.1 | 2.8 |
| 5 | 2.8 | 3 |
| 6 | 2.8 | 2.4 |
| 7 | 2.5 | 2.4 |
| 8 | 2.4 | 2.4 |
| 9 | 2.1 | 2.4 |

Figure M3: Mean Time Spent in NEW State

|  |  |
| --- | --- |
| Memory Schema 1 | 3.01 ms |
| Memory Schema 2 | 2.83 ms |

Figure M4: Mean Time Spent in NEW State Summary

### Conclusion

The mean time for processes spent in NEW state is less for Memory Schema 2 than Memory Schema 1. Since Memory Schema 2 admits more processes to the CPU than Memory Schema 1, we can conclude that Memory Schema 2 has a better memory management policy.

# Appendix A: Gantt Charts

A purple line with black text

Description automatically generated

Figure 1: Gantt chart of FCFS algorithm without I/O

A purple rectangular box with black text

Description automatically generated with medium confidence

Figure 2: Gantt chart of Round Robin algorithm without I/O

A graph with text and numbers

Description automatically generated with medium confidence

Figure 3: Gantt chart of Multi-level Feedback Queue without I/O