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

**Now assume that each process requests to do an I/O every 1 sec., and the duration of each of these I/O is 1 sec. Repeat part b).**

At

## 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

**Scheduler Report**

### Objective

The objective of this report is to test and compare the os kernal simulator’s different scheduling algorithms and how they perform whilst being subjected to a variety of different test cases. The outputs of the test cases will be utilized to calculate fields such as average throughput, average turnaround time, and finally average waiting time. From these calculated values, an analysis will be performed to help determine the strengths and weaknesses of each of the scheduling algorithms and how they perform compared to the others.

### Experiment Design

To perform the experiment a variety of test cases had to be developed to help demonstrate each of the different scheduling algorithms. The algorithms used in this experiment are as follows: First come first server, External Priorities, Round Robin. The list of scenarios below are the different test cases that the simulator was subjected to.

Each scheduling algorithm will be tested with at least 3 scenarios. One with processes that are mainly CPU bound, processes that are mainly I/O bound, and processes that are mixed in their variety of CPU and I/O workload. Each of these tests are different across all 10 scenarios although they are performing similar tasks with slightly different input, although it is meant to represent the same test when compared to the other schedulers. Ex: Scenario 0 (FCFS CPU Bound) will be compared with the similar but different Scenario 4 (CPU bound External Priorities). Below is the list of the different scheduler scenario test cases.

**List of different scheduler scenario test cases:**

1. Mainly CPU Bound Processes (FCFS)
2. Mainly I/0 Bound Processes (FCFS)
3. Mixed workload (FCFS)
4. Varying Process Priorities (External Priorities)
5. Mainly CPU Bound Processes (External Priorities)
6. Mainly I/0 Bound Processes (External Priorities)
7. Mixed workload (External Priorities)
8. Mainly CPU Bound Processes (Round Robin)
9. Mainly I/0 Bound Processes (Round Robin)
10. Mixed workload (Round Robin)

### Results

**CPU Bound**

**I/O Bound**

**Mixed workload**

|  |  |  |  |
| --- | --- | --- | --- |
| **Scheduler & Scenario** | **Average Throughput** | **Average Turnaround Time** | **Average Waiting Time** |
| **Scenario 0 (FCFS)** | 0.0285 | 58.33 | 13.33 |
| **Scenario 1 (FCFS)** | 0.0152 | 129.66 | 06.33 |
| **Scenario 2 (FCFS)** | 0.0162 | 131.66 | 11.66 |
| **Scenario 3 (Priorities)** | 0.0139 | 135.00 | 18.33 |
| **Scenario 4 (Priorities)** | 0.0214 | 113.33 | 40.00 |
| **Scenario 5 (Priorities)** | 0.0166 | 133.33 | 00.00 |
| **Scenario 6(Priorities)** | 0.0166 | 103.33 | 18.33 |
| **Scenario 7 (RR)** | 0.0240 | 94.33 | 39.66 |
| **Scenario 8 (RR)** | 0.0144 | 183.66 | 17.00 |
| **Scenario 9 (RR)** | 0.0157 | 138.00 | 21.33 |

**Figure S1**

**A graph of orange and grey bars

Description automatically generated**

**Figure S2**

**A graph with a red line

Description automatically generated**

**Figure S3**

### Conclusion

From the results on the table and displayed on the graphs there are a few trends which are clear. When discussing throughput, it is clear that the highest throughput occurs when the simulator is processed CPU bound processes, this makes sense as it doesn’t spend as much time with I/O. In graph 2 we can see that the first 3 scenarios on the bar graph are the CPU bound scenarios, with FCFS first, RR 2nd and External priorities 3rd. These findings make sense as both RR and external priorities can create situations where the CPU is waiting while a process is waiting, whereas this is much less likely with FCFS. The rest of the throughput graph demonstrates that it’s even across the board when dealing with I/O, this makes sense as with I/O bound processes the simulator is going to be held up in a similar way across all 3 different schedulers resulting in the similar throughput values we see.

From the results of the table and graph 1, we can observe that the fastest average turnaround time for a process in general results from the FCFS, whereas the slowest is from the RR. These results make sense as FCFS should allow processed to start and finish without being preempted other than for I/O, so for CPU bound cases FCFS should dominate on turnaround time, followed by RR, and external priorities in theory could have the same results as FCFS but it depends on which processes have the higher priorities, as that can affect who gets to run to completion first. Finally for waiting time, it can be observed that the worst waiting time results from RR, as processes are constantly switched in and with a time quantum of 1ms, this results in all the processes finishing at a similar time which creates a higher average waiting time. The waiting time for FCFS is the lowest in our simulation as FCFS only must wait in the ready queue while a process has arrived and is waiting its turn to execute or has returned from waiting and is also waiting its turn to execute. External priorities has very minimal waiting time when working with I/O bound as depending on the process priorities it can be structured to allow the process’ with largest I/O to run first so other processes can run whilst it waits.

Overall, the schedulers performed as expected throughout the tests and produced turnaround times, waiting times and throughput values that all make sense.

## 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