Assignment 3: Report Part C: Deadlock and Livelock Analysis

Esli Emmanuel Konate 101322259

Carleton University

SYSC 4001 : Operating Systems

Introduction

This report analyzes the execution of the TA Marking System I implemented in Part 2.a without the use of semaphores and in Part 2.b with the use of semaphores. We examine whether or not deadlock or livelock conditions happened during the execution and we also analyze the order of execution of the processes.

Test Metrics

The simulation makes use of various metrics of different amounts:

- Number of TAs: 3
- Number of Exams: 20 (Students from 1001 to 1020) + 1 termination file (student with 9999)
- Questions per Exam: 5
- Ran the tests on Ubuntu (WSL) on Windows

Execution Analysis

Part 2.a Analysis (Without the Semaphores)

Observations:

When running Part 2.a without the semaphores, all three TA processes were executed concurrently and processed all 20 exams correctly. The program also finished normally when we reached student 9999.

Observed Race Conditions:

Since we did not use semaphores in this part of the work, it was possible to observe multiple race conditions.

The first race condition was that related to the rubric. Multiple TAs read and corrected the same rubric at the same time. For example, TA 3 corrected Q1 from A to B, then TA2 corrected Q1 from B to C in a fast succession manner. This is a problem closely related to the “lost update problem”.

Second, this race condition was related to marking race condition. Multiple TAs often marked questions for different exams at the same time because there is no protection on the question status array.

Third, this race condition was related to the exam loading. When all questions were marked, multiple TAs had the possibility to try to load the next exam. The shared exam_idx variable was then incremented and there was no protection over that operation.

Deadlock and Livelock

No deadlock or livelock happened in Part 2.a. Because without semaphores, there are no ways of blocking operations that could lead to processes waiting for forever. Race conditions did happen, but processes continued executing instead of blocking.

Part 2.b Analysis (With the Semaphores)

Semaphores Used in the Simulation:

Semaphore	Objective/Goal	Starting Value
SEM_RUBRIC (0)	Mutex to modify the rubric	1
SEM_EXAM (1)	Mutex for loading next exam	1
SEM_QUEST (2)	Mutex to select questions	1

Observations:

There was a specific pattern to the execution:

First, All TAs review the rubric. When a TAs has to correct a question, it obtains SEM_RUBRIC, then it modifies the rubric, afterwards, it releases SEM_RUBRIC. Only one TA can modify the rubric at a time.

Second, TAs can acquire SEM_QUEST to select a question which was unmarked. Once the question is selected, its status is set to “being marked.” SEM_QUEST is then released and it allows other TAs to select different questions. After marking, TAs can obtain SEM_QUEST again to update status to “done.”

Third, When all the questions are marked, one TA gets SEM_EXAM, then loads the next exam, and resets all the current question status, and releases SEM_EXAM.

Deadlock Analysis:

There was no deadlock in Part 2.b,

Because Deadlock needs those 4 conditions:

Condition	**Did it Occur?**	**Why?**
Mutual Exclusion	yes	Semaphores give access for one at a time.
Hold and Wait	no	TAs never had multiple semaphores at the same time.
No Preemption	yes	Semaphores can't be taken forcibly.
Circular Wait	no	Semaphores were always obtained in the same order

Therefore, since Hold and Wait and Circular Wait were not present in this simulation, deadlock did not occur.

Livelock Analysis:

No livelock happened in Part2.b. Because of multiple preventative ways implemented.

The first way was that when no question is available but not all are done, the TA breaks out of the current marking loop and checks again. This allows the other TAs to complete their marking. In addition, the condition which was to check student numbers ensured that only the first TA who detected completion will load the next exam. While the others see the new student number and continue. Finally, the terminate flag is checked multiple times in order to have a good termination when we reach student 9999.

Conclusion

Both Part2.a and Part 2.b simulations were completed without deadlock or livelock occurring. Part 2.a has avoided deadlock by having no blocking mechanisms, but because of this, there were multiple race conditions and possibilities of having inconsistent data. While, Part 2.b avoided deadlock by having semaphores, which are obtained and released in small critical sections. TAs also never hold multiple semaphores at the same time, and sem_signal_op is always called after sem_wait_op no matter what happens. The simulation running with semaphores is the right way because it eliminates the risks linked to race conditions while also preventing deadlocks and livelocks.

KEY to the repository : https://github.com/J3Sli3/SYSC4001