SYSC 4001 Assignment 3 Part 2.c Report: Concurrency Analysis

Danilo 101297163

Aws

1. Analysis of Unsynchronized Execution (Part 2.a)

Deadlock and Livelock Conditions

The execution of the Part 2.a program (unlocked processes) did not result in a classic mutual deadlock (circular wait for resources). However, the program consistently failed to terminate cleanly, exhibiting a state of livelock or starvation in the final stages of the simulation.

The condition that led to this livelock was a non-atomic update to the shared state variables that control the TA's workflow:

- Cause (Missing Atomicity in Exam Marking Logic): The check for exam completion (questions_marked == NUM_EXERCISES) was performed locally by each TA process at the beginning of its marking loop. Since the mechanism for decrementing the questions_remaining count was local to each TA, a TA whose questions were successfully marked by a peer would enter a perpetual loop, checking questions and finding them all marked, but never updating its local state sufficiently to exit the loop and trigger the 'Load Next Exam' step.
- Result: Multiple TAs would remain active, continuously attempting and failing to mark a question, but none could make the required final state check to load the next exam (or terminate) correctly.

Race Conditions Observed

Before stalling, the execution of Part 2.a clearly demonstrated two primary race conditions:

1. Rubric File Corruption (Race on Writer): Multiple TAs were observed simultaneously printing messages indicating they were accessing the rubric.txt file for modification (writing). This led to inconsistent data being saved to the disk (e.g., losing increments to the rubric characters).
2. Question Marking Duplication (Race on Shared Flag): Multiple TAs could simultaneously read the shared memory flag for a question (e.g., Question 3), find it unmarked, and then proceed to both mark that question and spend 1.0–2.0 seconds simulating the mark, resulting in redundant work.

2. Analysis of Synchronized Execution (Part 2.b)

The implementation in Part 2.b utilizes System V Semaphores to enforce synchronization and resolve the issues found in Part 2.a.

Deadlock and Livelock Report

The synchronized program (part2b) ran successfully through all 21 exams (including the termination exam 9999.txt) and terminated cleanly. No deadlock or livelock conditions were observed in the execution.

Execution Order and Synchronization Proof

The synchronization mechanisms successfully regulated access to shared resources, allowing for high concurrency where possible and strict mutual exclusion when necessary.

1. Mutual Exclusion (Rubric Write): The RUBRIC_WRITE_SEM successfully ensured mutual exclusion for writing to the rubric.txt file. When a TA acquired the lock for correction, the console output showed a strict sequential boundary: TA 2 -> ACCESSING the rubric file (WRITE - SAFE). TA 2 <- FINISHED writing to the rubric file. No other TA initiated a write operation during this critical section.
2. Controlled Concurrent Marking (Dedicated Semaphores): The five dedicated semaphores (Q1_MARK_SEM through Q5_MARK_SEM) enforced integrity while maximizing concurrency.
    - Multiple TAs successfully marked *different* questions simultaneously (e.g., TA 1 marks Q4, TA 3 marks Q5).
    - If a TA attempted to mark a question claimed by a peer (after passing the sem_wait), the program correctly displayed detection and avoidance: TA 1 -> Question 5 already marked/claimed. Trying another. This proof confirmed that the concurrent readers-exclusive writer model was correctly applied to the shared marking status.
3. Atomic State Transition (Exam Loading): The EXAM_LOAD_SEM ensured only one TA handled the state transition upon exam completion. This prevented the index skipping observed in Part 2.a. The output showed the exam ID advancing cleanly (e.g., ID 2 -> ID 3 -> ID 4) with only one TA displaying the LOADING next exam (SAFE) message for each transition, guaranteeing consistent processing of the exam queue up to the final termination.