Assignment 3 (Part 2)

SYSC 4001: Operating Systems

Jonathan Gitej (101294584)
Atik Mahmud(101318070)

c) When running the programs there were no observable deadlocks or livelocks. In part 2 a there was no synchronization which leads to race conditions which means that TAs can modify the same shared values. In part 2 b SysV semaphores were used to implement synchronization which eliminates race conditions since now TAs cannot modify the same values at the same time. These semaphores control access to the rubric, loaded exams, marked questions and exam index. The process execution is as follows:

1. All TA processes are started concurrently.
2. Each TA:
    - Reads the shared rubric from memory.
    - Randomly decides whether to modify a line in the rubric.
    - If modification is needed:
        - The TA must acquire the semaphore, modify the line, write back to the file, then release the semaphore.
    - Marks questions on the current exam (only one TA can access a question meaning no TAs can grade the same question).
3. Once all questions of an exam are marked, one TA loads the next exam into shared memory.
4. This repeats until the last exam is reached (student ID 9999), at which point finished_exams = 1 signals all TAs to exit.

Due to semaphores only one TA is able to modify the rubric at a given time, also TAs also can't mark the same question which in turn avoids race conditions and improves efficiency.

Design Discussion in the Context of the Critical Section Problem

1. **Mutual Exclusion**
    - In our program, multiple TA processes share the rubric and exam information in shared memory.
    - To prevent simultaneous conflicting modifications, a SysV semaphore is used to protect access to the rubric.
    - Only one TA can modify a rubric line at a time, ensuring that race conditions do not occur when updating the rubric letters (A→B→…→Z).

This satisfies the mutual exclusion requirement because concurrent access to the shared resource is effectively prevented.

2. **Progress**
    - The system guarantees that if a TA wants to enter the critical section (e.g., modify the rubric), and the critical section is free, it will gain access.

The operating system scheduler, combined with semaphore operations, ensures that waiting TAs can eventually proceed, fulfilling the progress requirement.

3. **Bounded Waiting**
    - Each TA waits for access to the rubric for a limited time determined by the semaphore queue managed by the OS.
    - The design ensures that no TA is indefinitely postponed from updating the rubric once other TAs release the semaphore.