Design Discussion Assignment 3 Part 2

Chikezilim Afulukwe - 101279214
Abubakr Mohammed - 101287262

Design Discussion- SYSC 4001 Assignment 3 (Part 2)

This assignment required designing a system where multiple TA processes mark exams using shared memory. Since all TAs work on the same data, the main challenge was preventing race conditions and ensuring predictable interaction between processes.

Part 2(a): No Synchronization

Part 2(a) runs with no semaphores at all. Every TA freely edits the rubric, marks questions, and reads/writes shared memory. This leads to race conditions and unpredictable interleavings, since two TAs may update the same data at the same time. Although the output becomes unpredictable, no deadlock can occur because no TA ever waits on any shared resource. All TAs always make progress even if their operations overlap.

Part 2(b): Semaphore-Based Synchronization

In Part 2(b),to safeguard shared resources, we introduced semaphores in Part 2(b): • SEM_EXAM guarantees that only TA 0 loads the next test safely;
• SEM_RUBRIC guarantees that only one TA modifies the rubric at a time; and
• SEM_QBASE + q comprises five semaphores that each safeguard a single exam question.
This design allows TAs to work in parallel when possible(for example marking different questions at the same time) while still protecting critical sections. Since each TA holds only one semaphore at any moment and releases it immediately, the system stays organized and free of circular dependencies.

Design Discussion Assignment 3 Part 2

Critical Section Requirements

Mutual Exclusion: Semaphores are used to safeguard all crucial areas (exam loading, rubric writing, and each question). No two TAs can change the shared data at the same time.

Progress: A TA only waits when another TA is actively using the same resource. Since locks are released immediately and the system has no circular dependencies, at least one TA can always make progress. Once a TA finishes, other waiting TAs continue immediately.

Bounded Waiting: Each TA acquires and releases semaphores quickly, and no TA holds multiple semaphores at a time. Since the system handles waiting fairly, every TA gains access to a critical region within a finite amount of time.

Conclusion

All three of the essential section requirements are met by the semaphore-based design in Part 2(b), which effectively stops risky concurrent behavior. This results in execution that is steady, predictable, and devoid of deadlocks throughout several runs.