Carleton University

Assignment 1: Group Submission
Part 2 – C Report

SYSC 4001A - L3 - GROUP 8: Operating Systems

Student 1: Teereza Allaham (101289630)

Student 2: Sahil Todeti (101259541)

Due: December 1st, 2025

https://github.com/TeerezaAllaham/SYSC4001_A3P2.git

https://github.com/TeerezaAllaham/SYSC4001_A3P1.git

This assignment focuses on designing and implementing a marking system for TAs marking exams concurrently by using shared memory in a UNIX environment. The goal is to simulate a realistic marking system where several TAs work simultaneously to review a shared rubric, mark exam questions, and advance through a sequence of exam files until a designated sentinel exam is reached. In Part 2a, the system operates without any synchronization, intentionally exposing race conditions and unpredictable behaviour. Part 2b extends this by introducing system V semaphores to coordinate access to shared resources such as the rubric, question states, and exam loading, ensuring mutually exclusive access and preventing inconsistent updates. This demonstrates how interprocess communication and synchronization are essential in preventing concurrency issues and ensuring orderly progress in a multi-process environment.

In the semaphore-based version, the absence of deadlock is guaranteed by the structure of the critical sections. Each shared resource is protected by its own binary semaphore, no TA ever holds more than one semaphore at the same time. This prevents circular waiting, which is a necessary condition for deadlock. Additionally, all semaphore acquisitions (sem_wait) have corresponding releases (sem_signal), ensuring that no semaphore can be permanently held.

Livelock is avoided because progress always occurs. TAs eventually obtain the necessary semaphore, pick a question, complete the marking, and proceed to the next exam. The system's shared question states and exam index will always move forward, and the terminate flag ensures that all processes exit cleanly once the 9999 exam is reached.

In conclusion, based on repeated testing and the structure of the synchronization logic, neither implementation exhibited deadlock or livelock, and the controlled use of semaphores ensures correct and orderly progress among all TA processes.