SYSC4001 Assignment 3 - Part 2 (Report)

Authors: Aj Donald 101259149, Jayven Larsen 101260364

Part D report:

After executing the code written for part c, it appears that the semaphores lead to a deadlock state. Here are the factors why this is happening.

1. Resource Holding: Each TA acquires their first semaphore j and then attempts to acquire the second semaphore j+1. If j+1 is already acquired by another TA, the first semaphore is not immediately released until the second semaphore's acquisition fails.
2. Circular Wait Condition: Since there are 5 TAs and each TA tries to acquire semaphores in a circular manner (TA 1 tries to acquire semaphore 1 and semaphore 2, TA 2 tries to acquire semaphore 2 and semaphore 3, etc.), a situation can arise where:

TA 1 acquires semaphore 1 and waits for semaphore 2.

TA 2 acquires semaphore 2 and waits for semaphore 3.

TA 5 acquires semaphore 5 and waits for semaphore 1.

This creates a circular wait, where each TA is waiting for a semaphore that another TA is holding. In this case of my output of part c, all the processes are trying to access the database but arent able to, leading to an infinite loop of all the processes continuously trying to gain access to the semaphores

1. No Preemption: In this implementation, there’s no way of forcing a TA to release its first semaphore j until it determines that j+1 is unavailable. If all TAs are holding their first semaphore while waiting for the second one, none can make progress inevidavly creating the deadlock.

Part E report:

In the new solution trying to eliminate the deadlock situation, here are the key differences i’ve made in the code:

1. Ordered Semaphore Acquisition:
   * The acquire\_semaphores\_ordered function ensures semaphores are acquired in ascending order. This order will be the solution to the circular waiting issue we previously had. For example:
     + TA 1 acquires semaphore 1, then semaphore 2.
     + TA 5 acquires semaphore 1, then semaphore 5.
2. Simplified Semaphore Release:
   * I’ve also changed the way the semaphores are released. I’ve made it in the same order they are acquired to further eliminate the circular waiting issue.

The Deadlock prevention here is done by always acquiring semaphores in ascending order. With this and making sure the release of the semaphores are in the order aquired, circular wait is eliminated, and deadlock cannot occur.