School of Computer Science College of Science
Fundamentals of Operating Systems
homework 3 (Chapters 6&7)

---

1 Problems

1. Implement how a binary semaphore can be used to do mutual exclusion among n processes. The goal is to implement a solution that ensures mutual exclusion, meaning that only one process can access the shared resource at any given time. This is essential to prevent race conditions and maintain the integrity of the shared resource. You should provide followings in your code:

   - Implement a binary semaphore (using a language of choice that supports multithreading or multiprocessing).
   - Develop n processes, each of which will perform some operations in a critical section.
   - Use the binary semaphore to control access to the critical section, ensuring that only one process can execute the critical section at a time.
   - The processes should be able to enter the critical section, modify the shared resource, and exit the critical section without interference from other processes.

2. **Readers-Writers Problem**

   In a database management system (DBMS), multiple users may simultaneously read data from a database (readers), and some users may need to write data to the database (writers). The challenge is to ensure that multiple readers can access the database concurrently without conflicts, but when a writer wants to modify the database, it needs exclusive access to prevent data inconsistencies.

   Implement a program that simulates the readers-writers problem using synchronization mechanisms. Multiple readers and writers access a shared resource (e.g., file or a data structure like an array, ...). Readers can access the resource simultaneously, but writers must have exclusive access.

   Requirements:

   - Create a shared resource (e.g., a variable, data structure, or file) that readers and writers will access.
   - Implement functions for readers and writers to access the shared resource.
   - Use synchronization mechanisms (e.g., locks, semaphores) to ensure exclusive access for writers and allow concurrent access for readers.
   - Simulate multiple readers and writers accessing the resource.
   - Print relevant information (e.g., reader/writer activity, resource state) for each step to demonstrate proper synchronization and clearly comment your code to explain the use of synchronization mechanisms.

3. **Producer-consumer problem**
   Implement a solution for the classic producer-consumer problem with a shared buffer. You have a shared buffer that can hold a maximum of N items. There are two types of threads:
   Producers: These threads produce items and add them to the shared buffer. If the buffer is full, a producer should wait until there is space.
   Consumers: These threads consume items from the shared buffer. If the buffer is empty, a consumer should wait until there are items available.

   Thread Numbering:

   - Each producer and consumer should be assigned a unique number.
   - Print the number of each producer and consumer when they perform any action.

   Thread State Printing:

   - Print out the state of each thread whenever it performs an action (e.g., waiting, producing, consuming)
   - Clearly indicate the state transitions in your output.

   Termination Criteria:

   - Introduce a termination condition: terminate all threads when a total of K data items have been produced and consumed
   - Clearly specify the value of K (number of produced data), number of producers, consumers and N (Buffer size)

   Implement the solution using semaphores (or any appropriate synchronization mechanism) to ensure that:

   - No data races occur (i.e., multiple threads trying to access the buffer simultaneously).
   - The buffer does not overflow or underflow

   Clearly comment your code to explain the use of synchronization mechanisms.

   Example of output for (buffer_size =2,K = 5, number_of_producers=5 , number_of_consumers = 2):
   Producer 0, produced 56. Total items have been produced: 1
   Producer 1, produced 52. Total items have been produced: 2
   Buffer is full. Producer 2 is waiting...
   Buffer is full. Producer 3 is waiting...
   Buffer is full. Producer 4 is waiting...
   Consumer 0, consumed 56. Total items have been consumed: 1
   Consumer 1, consumed 52. Total items have been consumed: 2
   producer 3 has waited to push item : 12
   Producer 3, produced 12. Total items have been produced: 3
   Producer 0, produced 45. Total items have been produced: 4
   Consumer 0, consumed 12. Total items have been consumed: 3
   producer 4 has waited to push item : 57

Producer 4, produced 57. Total items have been produced: 5
Producer 1 is done.
Consumer 1, consumed 45. Total items have been consumed: 4
Producer 2 is done.
Consumer 0, consumed 57. Total items have been consumed: 5
Producer 4 is done.
Producer 3 is done.
Producer 0 is done.
Consumer 1 is done.
Consumer 0 is done