Topic: Synchronisation problems

Questions:

1.

(Readers-Writers problem; to do simple implementation using pthreads, and mutex locks for the CS region)

Create two threads.

Thread1 is a reader which reads a sample text file and transfers all the words in the file to a new file.

Thread 2 is a writer which appends a user provided message to the same file, n times to the files end

Use mutex locks to ensure synchronisation among the two threads.

Run your program and observe the result on each run.

Also remove the locks and observe the result.

2.

(First Readers-Writers problem; using pthreads, and mutex locks for the CS region)

Create five threads of which three are reader threads and two are writer threads. Each of the reader threads reads all the words of a common sample text file and writes the same to a new file (you can name the output files as one.txt, two.txt, and three.txt respectively from the from three reader threads). One of the writer threads appends "Hello world" message 10 times to the sample file, while the other writer appends "Hello everyone" message 25 times to the same sample file.

Ensure synchronisation by implementing your solution based on the "First-readers-writers problem".

Run your code as many times as possible and observe the results.

Also remove the locks and observe the result.

3.

Implement the Producer-Consumer-Bounded-Buffer Synchronisation problem using pthreads, and mutex locks for the CS region.

(You may consider simple integers as your array items and you may thereby use a bounded circular array as your buffer to hold the integer items produced by a producer thread and consumed by a consumer thread).

Run your code as many times as possible and observe the result for each run.

Also remove the locks and observe the result.

4. Implement the Dining Philosophers problem using pthreads and mutex locks. Run your code as many times as possible and observe the result for each run.

Also remove the locks and observe the result.

***