**Project 3 Concurrency: Final Report**

**Student: Mohamed Al Hamzy**

**Introduction**:

The problem being addressed here is the well-known readers and writers’ problem. Due to the need of multiple locks for data structure operations, the issue comes to play. Concurrent threads should be allowed to use the shared data. Only one thread has access to the critical section each time. Multiple readers can join simultaneously. However, due to the need to stall until writer threads are depleted this could cause some threads to starve.

**Code Explanation**:

A mutex lock does the job of delivering mutual exclusion for the threads as needed for our problem here. Two locks would be needed for this solution. The reader and writer will share one of them while the other would be exclusive to the writer threads. Adding an additional lock, shared by both reader and writer threads. Its purpose if to notify whenever the reader has finished using the resource. Giving priority for the writers.

The code can be tested with multiple scenarios by changing the scenarios.txt file contents. The version of the project I submitted is using the scenarios testing the edge cases given in the project instructions. Feel free to test the code out.

**References**:

* Github link in slides :

<https://github.com/remzi-arpacidusseau/ostep-code/blob/master/threads-sema/rwlock.c>

* Readers and Writers Lock Code from textbook figure 31.13 (Chapter 31)

After running scenario from instructions (“rwrrrrwrr wwrrrrwr”):

Text

Description automatically generated with medium confidence

**Pseudocode**:

Main function:

Opens scenario file

Reads threads from file

If the character encountered is a read thread then,

Create a reader thread using subroutine

If the character encountered is a write thread then,

Create a write thread using subroutine

Close scenario file

Join threads

Return from main function

Main function ends

Reader function:

Get shared lock

Get reader lock

Release shared lock

Perform critical section

Release reader lock

Reader function ends

Writer function:

Get shared lock

Get writer lock

Perform critical section

Release writer lock

Release shared lock

Writer function ends

**Conclusion**:

The solution posed provides a fairer chance to all threads. Allowing access to shared data using an extra lock to add security for the writer to enter the critical section. This still does not guarantee which threads run first. As that is a scheduler’s job.

Estimated work time: Around 16 hours