Project 3 Reader/Writer Locks

The main idea of this project is to find an optimal solution for the readers-writers problem in the way that does not starve readers or writers. That is, the optimal solution should allow multiple readers to read from a critical exclusive section but no writers are allowed to enter while reading process is taking place. On the other hand, no readers or other writers are allowed to enter an exclusive section if a writer is already granted access to the critical section.

To implement such solution, I used three different semaphores as the following:

```
sem_init(&sem->rd_mutex, 0, 1);          // readers mutex semaphore Initialized to 1
sem_init(&sem->rw_mutex, 0, 1);          // resource mutex semaphore Initialized to 1
sem_init(&sem->resourceQueue, 0, 1);     // resource queue mutex semaphore Initialized to 1
```

The main idea here is not to stave readers or writers, thus a resource queue semaphore is added to the previous established solution on the book to grant access to each thread based on FIFO while still allowing multiple readers to read at the same time, but once last reader waiting on the queue leave and if there is another writer take the turn, that writer will get access and exclude others waiting on queue threads (readers or writers) from getting access to such exclusive critical section until completely finish the writing process. That way readers and writers will be serviced as First In First Out mechanism.

Finally, I estimate I have worked on this project about 15 days allowed before deadline about 2 hours /day in approximately ~ 30 Hrs.

The readThread() Function Pseudocode:

```
A reader arrives to readThread()
threades wait at the resource queue to get queued to access the critical section
threads wait at another semaphore to access the critical section
if there is only one reader it stops the writer from access
signal for the next waiting thread in queue to execute (another reader)
readers can use the critical section now.
reading is performed ..Only readers allowed
wait the reader mutex and updating the reader counter
last reader leaving the critical section signal to a writer or another readers to access the critical section
signal for reader counter again
```

Project 3 Reader/Writer Locks

The writeThread() Function Pseudocode:

A writer arrives to writeThread()

threads wait at resource queue to get queued to access the critical section

threads wait at another semaphore to access the critical section

Writing is performed .. No readers or other writers allowed while execution

signal for the next waiting thread in queue to execute

signal to a writer or another readers to access the critical section

Three different scenarios have been used to test such implementation:

One scenario when writers and readers arrive on the queue interchangeably and get access respectively to their position on the queue.

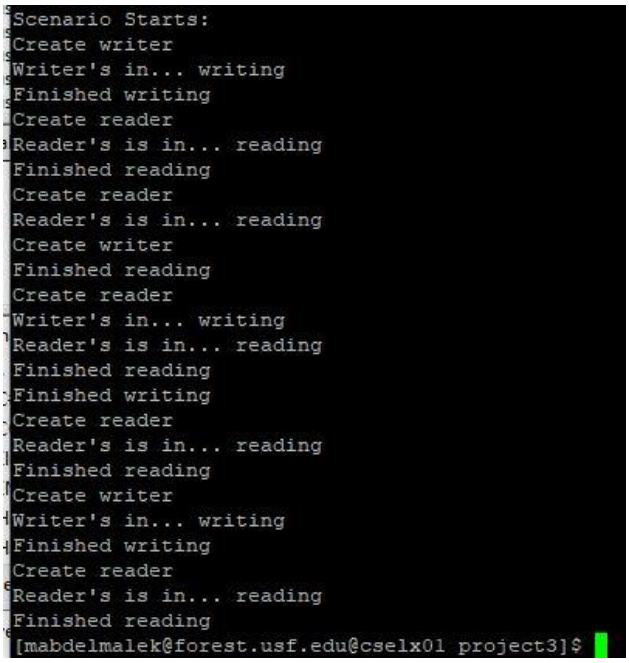
The following screenshot show the output when run the code:

```
Scenario Starts:
Create writer
Writer's in... writing
Finished writing
Create reader
Reader's is in... reading
Finished reading
Create reader
Reader's is in... reading
Finished reading
Create writer
Create reader
Create reader
Writer's in... writing
Reader's is in... reading
Finished writing
Finished reading
Reader's is in... reading
Finished reading
Create writer
Create reader
Writer's in... writing
Finished writing
Reader's is in... reading
Finished reading
Create writer
Writer's in... writing
Finished writing
Create reader
Reader's is in... reading
Finished reading
Create writer
Writer's in... writing
Finished writing
Create reader
Reader's is in... reading
Finished reading
[mabdelmalek@forest.usf.edu@cselx01 project3]$
```

Project 3 Reader/Writer Locks

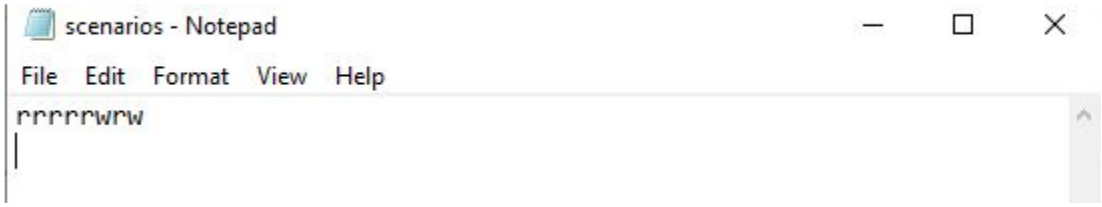
Second scenario when a writer arrive on queue and then bunch of readers arrive on a sequential manner and then another writer and then another sequence of readers and one last writer.

The following screenshot show the output when run the code:



Project 3 Reader/Writer Locks

Third scenario when a long sequence of readers arrive first and a couple of writers are at the end of the sequence which could have been starving till the end of all readers include readers behind them on the sequence finish reading before those writers could get granted access.



The following screenshot show the output when run the code: