MTL 458, Operating Systems, Homework 4

In this assignment, you are required to implement two versions of reader-writer locks. These will be used in situations where there is some shared data, and some users (called *readers*) want to read the data, while some other users (called *writers*) want to write into the data. The idea is to give access to other reader(s) when some reader(s) is(are) reading the data. On the other hand, the writer would need to have exclusive access to the data, where no other writer(s) or reader(s) would be allowed to access the data when it is being modified. A reader-writer lock is a special kind of lock where the user thread can specify whether it is a reader or a writer by requesting the corresponding lock using the functions *ReaderLock()* and *WriterLock()*.

As discussed in the class, there are two flavors of reader-writer locks. The first one is reader-preference lock, where a reader would be allowed to access the data even when a writer is waiting. Observe that while this is good for concurrency (many readers can access the data), but a writer may starve due to a steady stream of very small number of readers. The other version of the lock is writer-preference lock, where no more readers would be allowed to acquire the reader lock, once a writer is waiting for a lock to modify the data. You will be implementing a reader-writer lock in this assignment which can be used for both reader and writer preference locks.

Details:

(i) You need to complete the definition of the lock in the header file `rwlock.h`.

(ii) The function `InitalizeReadWriteLock()` must initialize the lock in the correct way.

(iii) The file `rwlock-reader-pref.c` should contain the reader-preference version of the functions. The functions that you need to implement are already given in the file. You need to fill in the correct code.

(iv) Similarly, the file `rwlock-writer-pref.c` needs to be filled-in with writer-preference version of the functions.

(v) Both the versions share the same header file, which contains the definition of the lock structure. So the lock structure is going to be common for both versions, but might contain elements which are used in only one version of the lock.

(vi) You are also given two additional tester programs that you may use to check the correctness of your implementation. I encourage you to read and understand what these programs do.

(vii) There are a couple of scripts (`run_reader_pref.sh` and `run_writer_pref.sh`) that run these programs with with some arguments. Inside the `.sh` files, the executables are followed by two arguments which represents number of reader and writer threads created. For example, `./rwlock-writer-pref 5 2` creates 5 reader threads followed by 2 writer threads and then lastly by additional 5 reader threads. These arguments can be modified for testing your code.

(viii) You can use the commands in these `.sh` files to compile and execute your code, and run it with different arguments to check the correctness.

(ix) You are encouraged to write your own test programs, that would test your implementation, by writing a new ones, or modifying the existing test programs given.

(x) The implementation will be evaluated by a tester code similar to one given to you, with some additional checks.

Guidelines: While doing the assignment, please be careful about the following.

- You will submit a `.zip` file `<Entry_No>_A4.zip` which would contain the files `rwlock.h`, `rwlock-reader-pref.c`, and `rwlock-writer-pref.c`, i.e., exactly the same as the names of the files that you need to modify. Please do *not* change the names.
- It's a relatively simple assignment, so the time given will be one week. You should start early as there will be no extension of the deadline.