CS304 Assignment № 2

Reader-Writer Locks

180010008 – Balsher Singh

Overview →

The objective of this assignment is to make us familiarize with writing programs with multiple threads, using locks and condition variables for synchronization across threads.

Consider an application where multiple threads of a process wish to read and write data shared between them. Some threads only want to read (let's call them "readers"), while others want to update the shared data ("writers"). In this scenario, **it is perfectly safe for multiple readers to concurrently access the shared data, as long as no other writer is updating it**. However, **a writer must still require mutual exclusion, and must not access the data concurrently with any other thread, whether a reader or a writer**. A reader-writer lock is a special kind of a lock, where the acquiring thread can specify whether it wishes to acquire the lock for reading or writing. That is, this lock will expose two separate locking functions, say, **ReaderLock()** and **WriterLock()**, and analogous unlock functions. If a lock is acquired for reading, other threads that wish to read may also be permitted to acquire the lock at the same time, leading to improved concurrency over traditional locks.

Here, there are 2 types of flavours of reader-writer locks:

1. Reader Preference → Reader is given preference over writer.

```
void WriterLock(struct read_write_lock * rw)
{
    pthread_mutex_lock(&rw->mutex);
    while(rw->reading !=0 || rw->wait_writer_num != 0)
        pthread_cond_wait(&rw->writers, &rw->mutex);
    rw->wait_writer_num++;
    pthread_mutex_unlock(&rw->mutex);
  //    Write the code for aquiring read-write lock by the writer.
}
```

Here, Writer will wait if any reader is reading. Thus, Reader preference.

2. Writer Preference → Writer is given preference over reader.

```
void ReaderLock(struct read_write_lock * rw)
{
    pthread_mutex_lock(&rw->mutex);
    while(rw->writing!=0 || rw->wait_writer_num > 0)
        pthread_cond_wait(&rw->readers,&rw->mutex);
    rw->reading++;
    pthread_mutex_unlock(&rw->mutex);
//  Write the code for aquiring read-write lock by the reader.
}
```

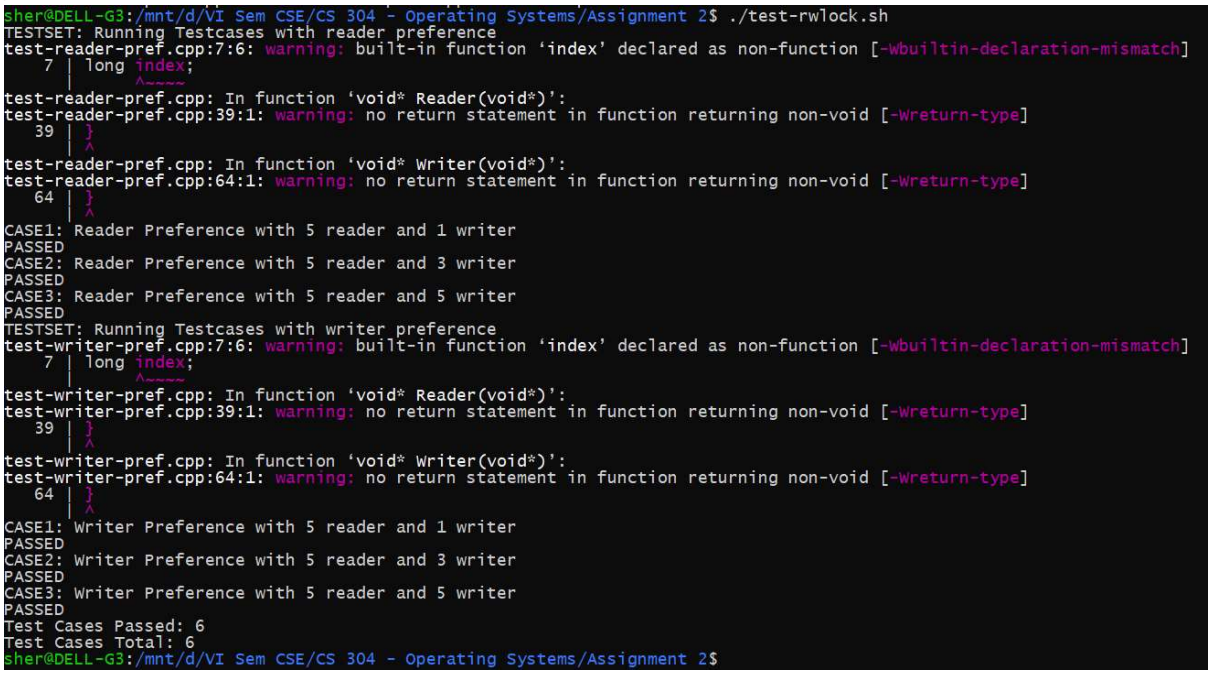
Here, Reader will wait if any writer is writing or any writer is in queue/waiting for writing. Thus, Writer preference.

Implementation →

- **rwlock.h**

The header file rwlock.h had to edited to add appropriate CV (Condition Variables), Locks and some variables pertaining to read write lock structure created.

```
struct read_write_lock
{
        pthread_mutex_t mutex;
        pthread_cond_t readers;
        pthread_cond_t writers;
        int wait_writer_num;
        int wait_reader_num;
        int reading;
        int writing;
};
```



- **rwlock-reader-pref.cpp & rwlock-writer-pref.cpp**

The rwlock-reader-pref.cpp is the file pertaining to reader preference and rwlock-writer-pref.cpp is the file pertaining to the writer preference. The functions had to be completed using Spin Locks & appropriate additions of CV with variables.

Test Cases →

Here, each process/person (reader/writer) is a different thread. So, scheduler may schedule any thread first. But next thread is executed according to the preference given either to reader or writer.

Test Case 1 : Reader Preference with 5 reader and 1 writer

- Assuming that the scheduler schedules Reader as the first process, so all the readers will finish reading only then writer will acquire lock to do modification to the shared file.
- Assuming if the scheduler schedules the writer process first and writer takes the lock, then after this 1 writer finish processing only then all readers can read at the same time.

Test Case 2 : Reader Preference with 5 reader and 3 writers

- Assuming that the scheduler schedules Reader as the first process, so all the readers will finish reading only then the 3 writers will acquire lock one by one to do modification to the shared file as at a time only one writer can write at a time.
- Assuming if the scheduler schedules the writer process first and writer takes the lock, then after this 1 writer finish processing only then all readers can read at the same time and after these readers read, the left over 2 writers can write one by one and finish their work.

Test Case 3 : Reader Preference with 5 reader and 5 writers

- Assuming that the scheduler schedules Reader as the first process, so all the readers will finish reading only then the 5 writers will acquire lock one by one to do modification to the shared file as at a time only one writer can write at a time.
- Assuming if the scheduler schedules the writer process first and writer takes the lock, then after this 1 writer finish processing only then all readers can read at the same time and after these readers read, the left over 4 writers can write one by one and finish their work.

Test Case 4 : Writer Preference with 5 reader and 1 writer

- Assuming that the scheduler schedules Writer as the first process, so all the writers will finish writing one by one only then readers will acquire lock to read from the shared file.
- Assuming if the scheduler schedules the reader process first and reader takes the lock, then all readers can read at the same time if scheduled concurrently and after these readers finish processing only then all writers can write to the shared file one by one and after all writers write, left over readers who were not scheduled concurrently first time can acquire lock and read.

Test Case 5 : Writer Preference with 5 reader and 3 writers

- Assuming that the scheduler schedules Writer as the first process, so all the writers will finish writing one by one, only then readers will acquire lock to read from the shared file. (Multiple readers can acquire lock at the same time and read)
- Assuming if the scheduler schedules the reader process first and reader takes the lock, then all readers can read at the same time if scheduled concurrently and after these readers finish processing only then all writers can write to the shared file one by one and after all writers write, left over readers who were not scheduled concurrently first time can acquire lock and read.

Test Case 6 : Writer Preference with 5 reader and 5 writers

- Assuming that the scheduler schedules Writer as the first process, so all the writers will finish writing one by one, only then readers will acquire lock to read from the shared file. (Multiple readers can acquire lock at the same time and read)
- Assuming if the scheduler schedules the reader process first and reader takes the lock, then all readers can read at the same time if scheduled concurrently and after these readers finish processing only then all writers can write to the shared file one by one and after all writers write, left over readers who were not scheduled concurrently first time can acquire lock and read.

Conclusion →

With all the cases confined above, we have demonstrated the variation of scheduling with controlled preferences over Readers and Writers.