Quyen Dang

Problem Approached:

Reader/Writer problem is one of common computing problems in concurrency. In the data sharing section, there are some processes only read data, and some processes only write data. Reader/Writer problem allows multiple reading processes but only one writing process at a time. The project purpose is using semaphore to design and implement the lock that non starving either reader or writer. That means our reader/writer lock would let all the readers get in (if there is in line) to read the data. When the writer would like to access data, it must wait until all the reader in the section done reading. Only one writer is allowed at a time. So, when there is a writer in the section, the other reader or writer need to wait until the writer done reading.

Solution:

I use two semaphores: one is the basic lock to control the read/write processes, and one to control the writer that allows only one writer but many readers at the same time. Both semaphores are initialized to 1. When reading process is created, the basic lock is decrement to 0 using sem_wait(&lock), and number of reader waiting is increased by 1. Then, I lock and decrease the semaphore writer by 1 using sem_wait(&wrt). I unlock the basic lock to let reader read data, and the semaphore lock now is 1. If another reader wants to access the data, the semaphore lock now is 0 using sem_wait(), so number reader is increased by 1 again. The process is repeated until no more reader reading data. If a writer tries to access data while other readers are reading data, the writer must wait until all reader finish reading. The writer would be wake by sem_post(&wrt) function when the number reader is 0, and the semaphore writer is 1 now. When the writer enters to the section, the writer writes to modify data. When writer done with writing, they left the section, and the semaphore writer become 1 by using sem_post() function.

Pseudocode:

Function for writer:

```
sem_wait(&wrt);
cnt = cnt + 1;
printf("Writer %d modified cnt to %d\n", (*((int *)wno)), cnt);
sem_post(&wrt);
```

Function for reader:

```
// Reader acquire the lock before modifying numreader
sem_wait(&lock);
numreader++;
if (numreader == 1)
{
    sem_wait(&wrt); // If this is the first reader, then it will block the writer
}
sem_post(&lock);
// Reading Section
printf("Reader %d: read cnt as %d\n", *((int *)rno), cnt);

// Reader acquire the lock before modifying numreader
sem_wait(&lock);
numreader--;
if (numreader == 0)
{
    sem_post(&wrt); // If this is the last reader, it will wake up the writer.
}
sem_post(&lock);
```

Time on project:

I spend about a week to learn the material and read the code from reference in about one week. And I code in about 2 days and fix the bugs in about 1 days.