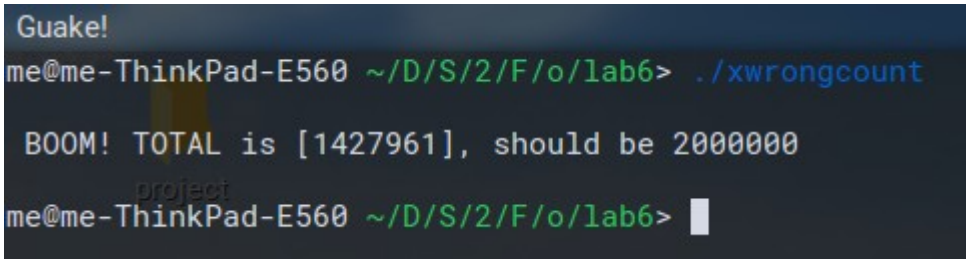
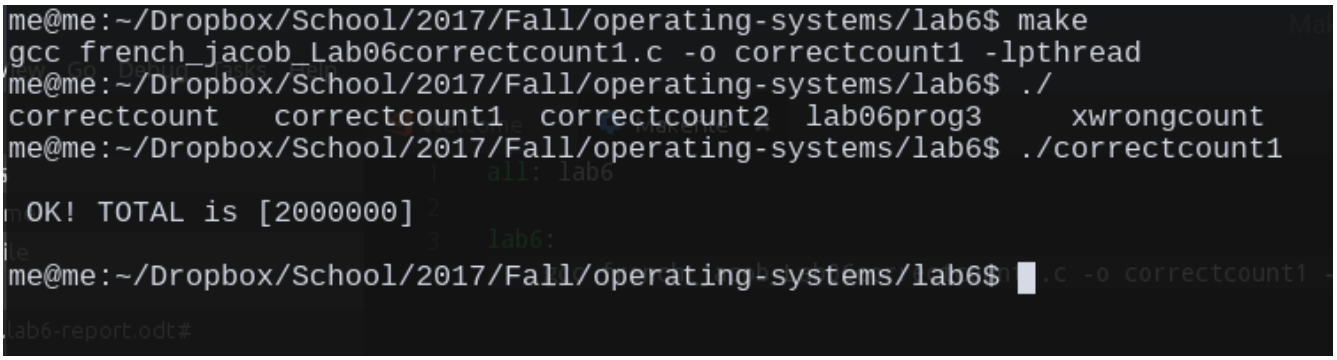
wrongcount.c



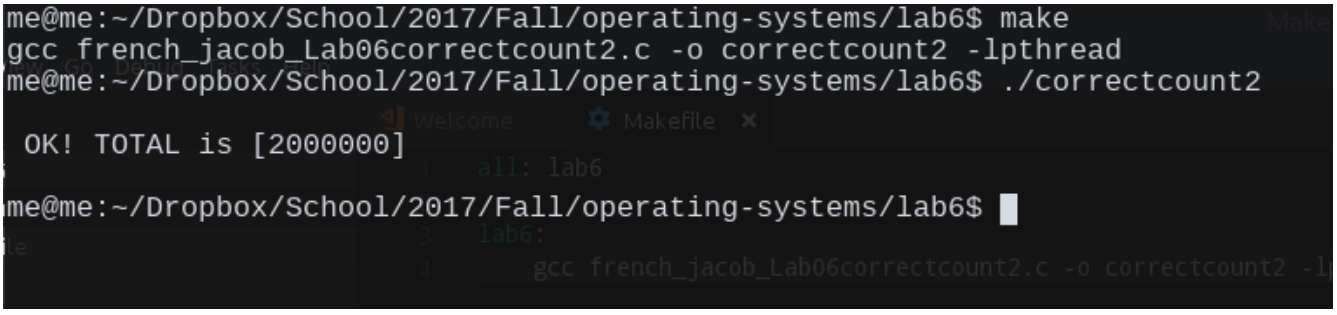
This program outputs the incorrect count because the threads are not synchronized.

french_jacob_Lab06correctaccount1.c



This program was synchronized through use of semaphores.

french_jacob_Lab06correctaccount2.c



This program was synchronized through use of a mutex.

french_jacob_Lab06prog3.c

```
me@me:~/Dropbox/School/2017/Fall/operating-systems/lab6$ make
gcc french_jacob_Lab06prog3.c -o lab06prog3 -lpthread
me@me:~/Dropbox/School/2017/Fall/operating-systems/lab6$ ./lab06prog3
threadID= 1505302272    i= 2    tmp= 1
threadID= 1513694976    i= 2    tmp= 2
threadID= 1496909568    i= 3    tmp= 3
threadID= 1488516864    i= 5    tmp= 4
threadID= 1409283840    i= 6    tmp= 5
threadID= 1400891136    i= 6    tmp= 6
threadID= 1392498432    i= 7    tmp= 7
threadID= 1384105728    i= 8    tmp= 8
threadID= 1375713024    i= 9    tmp= 9
threadID= 1367320320    i= 10   tmp= 10
me@me:~/Dropbox/School/2017/Fall/operating-systems/lab6$
```

1. Explain the difference between semaphores and mutex.

Mutexes and semaphores are both mechanisms used to control locking and unlocking of threads. They are similar to one another, but a mutex provides mutual exclusion to locking and unlocking threads whereas a semaphore is used as a more general approach to thread control. What this means is that a mutex can only be unlocked from the specific thread which has ownership. A semaphore, however, can be unlocked by any thread.

2. Explain another way to have a mutex, other than with the use of pthread_mutex

Another way to have a mutex is through use of a spinlock. A spinlock is a mutual exclusion mechanism like a mutex, but it continuously tries to unlock it's thread by using an infinite loop to check the lock variable rather than waiting for a signal from the thread.

Conclusion:
This lab showed that there are different ways of synchronizing threads. Thread synchronization is useful because it allows the programmer to specify which files and resources can be shared between processes. Using threads in this manner allows for better management of resources which can help to make programs more efficient.