Name: Kent Mark
Section: 5
University ID: 583972417

CprE 308 Lab 3 Report

Summary (20 pts):

By doing this lab I was able to gain a better understanding of concurrent programming using pthreads. In the first part of the lab I learned how to create a multithreaded program along with the uses of the create() and join() calls. Then I learned how to use mutex, more specifically the lock and unlock commands. Lastly, I learned about condition variables how they can be used to synchronize the threads.

Lab Questions:

3.1:

10 pts Add a sleep(5) statement in the beginning of both thread1 and thread2. Compile and run the program. Do the messages get printed? Why or why not?

The messages don’t get printed because the sleep statement makes it where only the main function will run while the two threads are still “sleeping”.

5 pts Add two pthread_join() calls to join t1 and t2 just before the print statement in main(). Compile and run the program. Do the threads' messages get printed? Why or why not?

The threads messages get printed because the main function now has to wait for the two threads to finish before it can print it's message.

5 pts Include your code with comments explaining the usage of *pthread_create()* and *pthread_join()*.

```c
#include <stdio.h>
#include <pthread.h>


void * thread1();
void * thread2();


int main(int argc, char* argv)
{
        /*declare thread variables*/
        pthread_t t1;
        pthread_t t2;

        /*creation of threads 1 and 2*/
        pthread_create(&t1, NULL, thread1, NULL);
        pthread_create(&t2, NULL, thread2, NULL);

        /*wait for threads 1 and 2 to complete*/
        pthread_join(t1, NULL);
        pthread_join(t2, NULL);

        /*identify self as main process*/
        printf("Hello, I am the main process\n");

        return 0;
}
```

```c
void * thread1()
{
        printf("Hello, I am thread 1\n");
}

void * thread2()
{
        printf("Hello, I am thread 2\n");
}
```

3.2:

3.2.1:

5 pts Compile and run t1.c. What is the output value of v?

V = 0

15 pts Delete the *pthread_mutex_lock* and *pthread_mutex_unlock* statements in both increment and decrement threads. Recompile and rerun t1.c. What is the output value of v? Explain why the output is either the same as, or different than, before.

V = -990. The output is different because the thread is no longer locked from increasing or decreasing.

3.2.2:

20 pts Include your modified code with comments labeling what you added or changed.

```c
void again() {
    pthread_mutex_lock(&mutex);

    /* again thread waits until done == 2 */
    while(done != 2)
        pthread_cond_wait(&done_world, &mutex);

    printf("Again!");
    fflush(stdout);
    pthread_mutex_unlock(&mutex); // unlocks mutex
    return;
}
```

3.3:

20pts Include your modified code with comments labeling what you added or changed.

```c
void *producer(void *arg)
{
  int producer_finished = 0;

  while (!producer_finished)
  {
    pthread_mutex_lock(&mut);
    while (supply != 0)
      pthread_cond_wait(&producer_cv, &mut); //wait to produce

    if(num_cons_remaining == 0) //stop producing if there are no consumers
    {
      printf("There are no more consumers, producer is finished\n");
      fflush(stdin);
      return;
    }

    printf("\n");
    fflush(stdin);

    supply += 10; //Produce 10 items
    pthread_cond_broadcast(&consumer_cv);

    pthread_mutex_unlock(&mut);
  }
  return NULL;
}
```