CSCD 340
Lab 5

In class we discussed pthread code. Here is a small summary of the functions we need.

PTHREAD FUNCTIONS

First, you will need to be able to create new threads, and you can do just that with the following PThread API routine:

- int pthread_create(pthread_t *thread, pthread_attr_t attr, void* (*start_routine)(void*), void *arg)

The first argument, pthread_t *thread, allows the calling thread to keep a structure that contains data relevant to the thread it creates; you can think of it as keeping a reference. We will not use the pthread_attr_t attr argument. The start_routine argument is a function pointer to the function that the thread will start in. In C you will always write a function name for this argument. Finally, the arg argument is a value of ambiguous type (thus the void * typing) that will be passed to the start_routine that is called when the thread begins.

A typical call to create a thread looks like:

```
pthread_t  the_other_thread;
pthread_create(&the_other_thread, NULL, startingFunction, NULL);
```

The call above creates and runs a new independent thread of execution which starts at the top of startingFunction.

Other pthread functions you might need are:

- int pthread_join(pthread_t thread, void **value_ptr);
- void pthread_exit(void *value_ptr);

pthread_join - When thread A joins thread B, thread A will not continue until thread B has completed and exited.

The *pthread_exit* function terminates the calling thread and makes the value value_ptr available to any successful join with the terminating thread.

TO COMPLETE

1) In class we also looked at some basic threaded programs. Emulate that code except create 6 threads where threads 1, 3, 5 call a simple hello function, and where threads 2, 4, 6 call a simple good bye function.

   Make sure that you join the threads outside the for loop where the threads were created.

   Name this pthreaded C program named lab5Prob1.c. Run this program 4 times and capture the output in a PDF named lab5Pthreads.pdf.

2) One thing we talked about in class but I never showed the PThread code was the concept of each thread having its own copy of stuff. Let's create a program to illustrate this

   Create lab5Prob2.c - you will need to create a main that is passed argc and argv.

   int main(int argc, char *argv[])

   When the program is run 2 command line parameters will be passed, both will be integers. In main complete a simple check to ensure that you have the 2 command line parameters. If you don't have two command line parameters display an error message and exit.

   Here is a call to the executable ./a.out 10 5

   This will create 5 threads and a 1D array that is 10 elements. The array will be filled with the values starting at 100 and increment until the last element array[0] = 100, array[1] = 101, etc.

   You need to make sure that you print out exactly what is going no. Example: thread 1 filling array element 0 with 100.

   Once all the threads have completed and the array is filled, print out the contents of the array.

   Don't forget you will need to compile with –lpthread (that is a lower case L)

   Compile, run and save your output from the run as lab5Pthreads.pdf. Include in that PDF the answers to the following questions. I expect thoughtful responses (one word or non-thoughtful answers will earn 0 points).

   a) When you create a thread you use the command pthread_create(&tid[x])
      What does the value in the tid represent? NOTE: the answer is not the value of x. I am expecting you to explain to me why create needs this value and what it truly represents.
   b) What is the purpose to the call to pthread_join?
   c) Is the call to pthread_join required? Why or why not?
   d) When you call pthread_join you pass the tid, what does that tid represent?
   e) Is that tid the current threads tid or can you pass it any tid?

3) In class we discussed the producer and consumer problem and wrote the code.

   Below I have provided a portion of the code for the producer. Your task is:

   - Create lab5pc1.c
   - Modify the producer code by adding printf statements to illustrate what is going on. For example: Producer creating widget 1234, and placing it in the buffer or buffer is empty consumer is blocking.
   - Add the appropriate consumer code
   - Compile and run the code and save the output – Just enough to illustrate what is happening.

- The producer code is:

```
for(x = 1; x <= MAX; x++)
{
        pthread_mutex_lock(&the_mutex);
        while(buffer != 0)
        {
               pthread_cond_wait(&condp, &the_mutex);
        }// end while

        buffer=x;
        pthread_cond_signal(&condc);
        pthread_mutex_unlock(&the_mutex);
}// end for
```

- In lab5pc.pdf answer the following
    - In your own words explain what pthread_cond_wait does (I don't want a copy of the man page – hence in your own words)
    - Why is the first parameter condp instead of condc for the producer and condc instead of condp for the consumer?

4) In main there are two lines similar to:

```
pthread_create(&pro, 0, producer, NULL);
pthread_create(&con, 0, consumer, NULL);
```

Switch them and rerun the program. Name this file lab5pc2.c. Run the code and save the output in PDF file.

As part of the output add a comparison of #3 as compared to #4, and answer the following.

- Did switching those two lines make a difference? Hint: if your answer is no, then you don't have enough and/or the proper printf statements to illustrate what is happening. There is a difference, and I am looking for a thoughtful explanation of what is going on here. Also comment on if you need to change the destroy order at the end of main.

5) In #3 the producer produces and the consumer consumes for MAX (100) times. Modify the code so there is a true buffer that can potentially fill up.
- Each for loop will be replaced by an infinite while loop
- Buffer will need to be modified so it is an array of size MAX.
- Name your file lab5pc3.c
- Compile and run your program and capture the output save it in lab5pc.pdf
- Remember the program runs for ever so let it run for a few seconds and then kill it.

6) Using #5 as a starting point add code to main to create a different number of producers than consumers.
- Name this file lab5pc4.c
- Compile the code, run the code and save the output in lab5pc.pdf.
- As part of the output write at least one paragraph commenting on the behavior.
    1. Was the behavior expected?
    2. Where there problems?
    3. Did the producer/consumer keep up or was there a lot of sleeping?

Note: a paragraph is defined as a minimum of 4 complete sentences, 12 pt font, 1 inch margins

TO TURN IN

A zip

- Your C files
- A Makefile with targets for all and each individual problem (prob1, prob2, prob3, prob4, prob5, prob6, all)
- Your PDFs – Ensure you label each question in the PDF

You will submit a zip file named your last name first letter of your first name lab5 (Example steinerslab5.zip)