2threads-api-lab

CST 334 (Operating Systems)
Dr. Glenn Bruns

Lab: Threads API

1. Copy the files in

```
/home/CLASSES/brunsglenn/OSTEP/HW-Threads-RealAPI
```

on mlc104 to a directory of your own. (Alternatively you can get the code from the OSTEP page.)

Use command "more" to examine the contents of the file 'mythreads.h'. It defines functions just like pthreads functions, but with simpler error handling. The functions in mythreads.h are just like pthreads calls, but start with capital 'P', and do simple error-checking.

2. Make sure you can compile and run the code:

```
gcc -o main-deadlock main-deadlock.c -pthread -Wall
```

3. Look at the code of main-deadlock.c and see if you can find anything wrong with the code (think about the counter program we saw). Spend time understanding the code.

Can you add print statements to illustrate the problem? The name suggests the code will deadlock. Does it always deadlock? If not, how often does it?

4. Examine `main-deadlock-global.c`. Does it have the same problem that `main-deadlock.c` has?

5. If you still have time, do problem 6 at the end of Chapter 27. Quoting from the text:

Let's next look at main-signal.c. This code uses a variable (done) to signal that the child is done and that the parent can now continue. Why is this code inefficient? (what does the parent end up spending its time doing, particularly if the child thread takes a long time to complete?)

6. If you still have time, do problem 8 at the end of Chapter 27. We haven't talked much about condition variables, so mostly focus on trying to read and understand the code. Look at Section 27.4 of the text if you want.