Operating Systems (coe628) Lab 5

Week of February 13 2017

Duration 1 week

Objectives

- Learn how to use threads in C (using the POSIX `pthread` package)

Preparation

- Familiarize yourself with the POSIX thread library: http://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html#BASICS. You will use the following calls:
  - `pthread_create`
  - `pthread_join`
  - `pthread_mutex_lock`
  - `pthread_mutex_unlock`
  - `pthread_self`
- You will also use the function `getpid()`

Requirements

When two or more threads access a shared variable at the same time a *race* condition may occur; this can affect the program correctness.

When two or more threads try to access a shared variable, the scheduler may switch from the active thread to another while the thread has not finished working on that variable. This may result in incorrect flow of execution.

In this exercise, you will create a situation where race condition occurs. You will implement a program where counter threads increment a shared variable. Since the counter threads do not have exclusive access to the counter variable when accessing it, the race condition happens and the variable is not incremented correctly.

1. Download the template Netbeans project. Click here.
2. In the code, each of the placeholder comments should be replaced with one or more instructions in order to complete the program. The required libraries have been added to the file but you may need to include more libraries if you follow a different approach. In the source file the function `count` increments the shared variable `globalNumber` by one for 10 times. The program creates five threads that run the counter function and therefore at the end, the value of the `globalNumber` variable should be 50. Read the counter function to see how it works.
3. The code you download should compile and run. It should print that the counter is 0 (zero) since

no treads have been created.

4. The program should make sure the threads have finished before printing the value of `globalNumber`. Replace placeholder B with a set of calls to `pthread_join` to make sure the threads has finished working before printing the value of the `globalNumber` variable.
5. Your program should now work. Run it and check if the final value of the counter is equal to 50. Try to figure out how the change of execution flow from one thread to another can affect the counter variable.

And Finally: Submit your lab

To submit your lab do:

1. Zip your project into a file called lab5.zip. i.e. zip -r lab5.zip lab5
2. Submit the zip file with the command: `submit coe628 lab5 lab5.zip`

That's all folks....