3locks-lab

CST 334 (Operating Systems)
Dr. Glenn Bruns

Lab: locks

The purpose of this lab is to give you experience with using locks in multi-threaded code.

1. On the hosting server, copy ‘counter.c’, 'Makefile', and ‘mythreads.h’ from

    `/home/CLASSES/brunsglenn/cst334/labs/locks`

    to a directory of your own.

2. Read the Makefile. Then compile and run counter.c. Do you get exactly 20 million as the output? Run the program many times to see how often you get 20 million as output.

3. Look at file mythreads.h. It is provided by the authors of our text to make it a little simpler to write pthreads code. Be aware that we will not always use this file.

4. Copy counter.c to counter-with-locks.c. Then edit counter-with-locks.c so that the statement that increments the counter is surrounded by lock() and unlock() calls. Refer to the lecture slides as needed, but enter code by hand, don't copy/paste.

5. Modify the Makefile so that it includes a target to compile counter-with-locks.c

6. Run the counter-with-locks code many times. Do you always get 20 million as output now?

7. Can you notice a slowdown in your code after adding the locks?

8. Copy counter-with-locks.c to shuffle.c. Then modify shuffle.c in the following way:

    ○ change it so that four threads are created and joined instead of two

    ○ pass "A" to the first thread, "B" to the second, "C" to the third, etc.

    ○ remove everything having to do with the counter variable (in other words, the first thread will just print "begin A" and then "end A")

    ○ remove everything in the mythread() function except the two print statements and the return statement

9. Modify the Makefile so that it includes a target to compile shuffle.c. Run shuffle.c and see what you get. Do you ever get a 'begin' for one thread that is not followed by 'done' for the same thread?

10. Modify shuffle.c so that this statement is inserted between the two print statements in mythread():

```
usleep(100);
```

This causes the calling thread to delay for 100 microseconds. You will need to also add `#include <unistd.h>` at the top of your file.

11. Compile and run shuffle. Do you get the same output as before?

12. Now edit shuffle.c to surround the print statements in mythread() with pthread lock() and unlock() calls. Compile and run shuffle. Do you get the same output as in the last step? Why? Do you always get the same output?

13. If you still have time, figure out how to time your code to see how long counter-with-locks takes to run compared to counter. Hint: use the 'time' command. See man page for details.

14. If you still have time, write a bash script and awk code to see, in step 2 above, how often you get 20 million if the code is run many times.