# Implementing Peterson’s Algorithm in pThread and Mutex

Note: Please use text highlight color to highlight your answer.

1. Open the file **pthreadcounter0.c**.
   1. In the line #18 of **pthreadcounter0.c**, there is a comment. Delete the comment sign ‘//’ and recompile pthreadcounter0.c. Please show what you have observed (paste the screenshot).
   2. Observe the output. Is the printout appropriate or synchronized?
   3. Use the knowledge you have learned in Peterson algorithm. Implant the Peterson algorithm to threads based on pthreadcounter0.c. Copy pthreadcounter0.c to pthreacounter2.c and make sure the modification is on pthreadcounter2.c. Here is the requirement: (1) Shared variables are flag0, flag1 and turn. (2) Each thread has its own function to run, named trythis0(…) and trythis1(…) respectively. (3) Use the Peterson algorithm. (4) The critical section is in pthreadcounter0.c the lines #15-#18.

Compile the program and run it. Please show your code and paste the screenshot to see if the printout is as you expect.

1. Read the source code, pthreadmutex0.c, and answer the following questions (<https://randu.org/tutorials/threads/>).
   1. What is the purpose of pthread\_mutex\_lock()?
   2. Compile the source code and run. Paste the screenshot of running results.
   3. The variable shared\_x is changed by threads. Can processes share variables? What is the difference between thread and process.
   4. Does pthread mutex lock need to initialize before being used? What function is it invoked?
2. Copy the source code pthreadcounter0.c to pthreadcounter3.c. Use the pthread\_mutex() to modify pthreadcounter3.c, so that it implements the thread synchronization, that is, the variable named counter can be synchronized (lines #15-18 in pthreadcounter0.c are the critical section). Additionally, the two threads should keep competing for the critical section until it runs away from for loop.

Please show your code and paste the screenshot to show the results.