COMP 3430 – Operating Systems – Lab 2
Introduction to Concurrency

Purpose

The goal of this lab is to introduce you to a real-world threads facility (pthreads), to give you initial hands-on experience with very simple synchronization using pthreads and to have you explore some other facilities for synchronization under Linux.

Introduction

Your lab will again take place in E2–420 EITC (where you will physically sit to do the lab) and will again use the Mac machines via remote login to aviary using SSH.

How to Prepare for the Lab

Your first step in preparation for the lab should be to read over the lab problems described below and make sure you understand them generally. The lab demonstrator will be available to answer specific questions during your actual lab. Your second step should be to review any relevant background material (e.g. the class notes, handouts, and relevant links) as well as to read any man pages specified in this document. Additionally, you might want to download the code in the file `threadeg4.c` from the course homepage in advance and make sure you understand it. Your final step in preparation for the lab should be to quickly review the relevant notes covered in class, to make a note so that you remember to bring the notes with you to your lab (since they may be useful), and to print a copy of this lab document and the hand–in document (if you have not already done so) to bring with you to the lab.

What to do?

Answer the questions below on the hand–in sheet. Be sure to hand in your completed sheet to the demonstrator before you leave the lab. This is how you will be graded!

Pthread mutex variables

The Linux pthreads library provides mutex variables. These permit threads to provide mutual exclusion on their critical areas. Consult the man pages for `pthread_mutex_init()` and `pthread_mutex_lock()` and answer the following questions.

Questions

[2] 1a. In Linux, there are two ways to initialize a p-thread mutex variable with default attributes. Briefly describe them. (Note: the second way requires the variable to be statically allocated.)

[2] 1b. Once a mutex variable has been destroyed using `pthread_mutex_destroy()`, can it be reused by the thread? Explain.

[3] 1c. What does the `pthread_mutex_trylock()` function do? When would this function be useful?

2. Download the program `mutex_recursive.c` from the Examples page. The program attempts to lock a mutex multiple times.

[3] 2a. Compile and run the program without any changes. What happens? What output do you see? Can you explain why this program does not execute successfully? Note: you will have to terminate the program using ctrl+C.

[3] 2b. Consult the man page for `pthread_mutex_lock()`. What type of mutex can be successfully used the way this program attempts to do? Make a change in the program to set the mutex to the correct type, then compile and run the program. What results do you obtain this time? Can you think of an application that would need this type of mutex?

3. Download, compile and run the program `threadeg4.c` from the Examples page. The program creates 20 threads, all of which are accessing a shared `int` variable.

[1] 3a. Examine the program. What should be the value of the shared variable at the end of the program?

[1] 3b. Run the program 10 times and record the values printed for each run. How can you explain the results observed?

[5] 3c. Rewrite the program by adding a mutex variable that the threads can use to enforce mutual exclusion on their critical section. Be sure to initialize and destroy it in the main function. Run the program 10 times and record the values printed for each run. Do the values agree with your answer from question 3a?

Demonstrate your programs for the lab demonstrator. The demonstrator will initial and make any comments on the hand–in sheet. If you are not able to demonstrate a working program during lab time then notify your TA and when finished submit your completed program using the dropbox on the D2L by the end of the day – if you do demonstrate, then a dropbox submission is not necessary.