CSE3321.3
Assignment 2.

Using Threads.

Due Date: Wed. Nov. 15 4:00pm.

1. POSIX threads

In all the following questions include a short answer and on which man page you found the information.

(1) What is the naming conventions for pthread function calls?

(2) Enumerate the synchronization mechanisms for POSIX threads?

(3) Why is it a good programming practice to call `pthread_cond_wait` from within a loop?

(4) Which thread executes after a thread is woken up from a condition variable.

(5) What happens if a thread blocks on a condition variable while holding two mutexes.

(6) What is a detached thread.

2. Probabilistic Analysis of Pthreads.

In this assignment you will write two pthreads programs that evaluate some aspects of the pthreads system. The first and simpler of the two, lets threads write to the same file using mutexes to avoid race conditions. The main thread creates `N` (`N` is an optional command line argument by dafault 100) threads and all of them immediately block on condition variable `start_line`. After all threads are created, the main thread wakes them up and then waits until they all terminate with `join`. The threads after waking up lock the mutex `filemut` and write their PID followed by newline to file `pthread_stats` and then release the mutex. After they do this `M` times each (the second optional command line argument that defaults to 1000) they `join` the main thread.

In the second pthread program of the assignment the main program creates `N` threads designated as `red` and then another `N` threads designated as `green`. All of them immediately block on condition variable `start_line`. After all threads are created, the main thread wakes them up and then waits until the all terminate with `join`. The threads after waking up lock the mutex `filemut` and write their PID followed by the word `red` or `green` followed by newline to file `pthread_stats`. The difference in this second version is that all the `red` threads write first their PID `M` times releasing and re-acquiring the mutex, then all the `green` ones another `M` times, then the `red` threads again, and so on, until they have written `L` (another optional command line argument that defaults to 10) times each.

Both programs accept optional arguments. The first one is `N`, the second one is `M` and the third one (applies only to the second program) is `L`.

To do the analysis write a small program in C, AWK, or python (I strongly recommend AWK but python is also OK) to calculate the mean and standard deviation of the "runs". A run is the number of consecutive writes by the same thread.