Prof. Jingke Li (FAB 120-06, li@cs.pdx.edu); Class: MW 2:00-3:50pm @ FAB 40-07; Office Hr: MW 1-2pm & by appt.

Lab 2: Programming with Pthreads

Download and unzip the file `lab2.zip` from D2L. You'll see a `lab2` directory with some program files.

Exercise 1: Condition Variable

The program file `condvar-pthd.c` is an incomplete Pthreads program. The `main()` routine creates two threads, one sender and one receiver. Complete the program by providing code for these two threads, so that the sender will send a signal, and the receiver will wait for the signal.

The `sleep(1)` call in `sender()` is to help to see the waiting. When you run the completed program, you should see two message lines first:

```
Sender starts ...
Receiver starts ...
```

After a pause, you should see the third line:

```
Signal received!
```

Exercise 2: Race Condition

The program file `arraysum-pthd.c` is a Pthreads version of a array sum program. Use an editor to open the file; read and understand the program; and then compile and run it:

```
linux> make arraysum-pthd
linux> ./arraysum-pthd 1000 10
...
The sum of 1 to 1000 is 500500
```

1. Try the program with different array sizes and number of threads. What is the sum of 1 to 12345?
2. Comment out all the occurrences of Pthreads locks in the program, and re-compile it. Now, this program has the potential for race conditions. Run the program until you see an evidence of a race condition occurring. Write down the array size, the number of threads, and the evidence.

Exercise 3: Work Partition

The program file `mtxmul.c` contains a simple sequential implementation of matrix multiplication. Use an editor to open the file; read and understand the program; and then compile and run it.

1. Convert `mtxmul.c` into a Pthreads program, `mtxmul-pthd.c`. Use `N` threads, one for each iteration of the `i` loop. Compile and test your program.
2. Write a second version of the Pthreads program, `mtxmul-pthd2.c`. In this version, the number of threads is not fixed. The program reads in an optional command-line argument representing the number of threads. If the argument is not provided, the program use the default value of `N`. For example,

   ```
   linux> ./mtxmul-pthd2 4    // using 4 threads
   linux> ./mtxmul-pthd2      // using the default of N threads
   ```

   Use a simple partitioning scheme to partition the workload for the threads.