Prof. Jingke Li (FAB 120-06, lij@pdx.edu); Class: MW 14:00-15:15 @ FAB 47; Lab: F 10:00-11:30 @ FAB 88-10.

Lab 2: Shared-Memory Programming with Explicit Threads

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

1. The Pthread Sum Program

The program file `sum-pthd1.c` is a simple Pthread program for computing the sum of `N` values. (We've shown this program in class.) Read and understand the program; and compile and run it:

```
linux> make sum-pthd1
linux> ./sum-pthd1
```

Exercise In class, we showed additional versions for this program. Follow the slides and create a new version, `sum-pthd4.c`, that corresponds to Version 4 of the slides. it should have the following features:

1. Worker thread k is assigned to run on CPU k. (Using a round robin arrangement if there are more threads than CPUs.)
2. The printout contains both worker's ID and CPU ID.
3. Parameters `N` and `P` can be entered at the command-line.

Compile and run this new program.

2. Condition Variables

The program file `condvar-pthd.c` is an incomplete Pthread 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 two more lines:

```
Signal sent!
Signal received!
```

Exercise Write a second version of this program, `condvar-pthd2.c`. The new program has two threads running the `receiver()` routine. Give each receiver thread an ID, so that you will see messages indicating their thread:

```
Sender starts ...
Receiver 1 starts ...
Receiver 2 starts ...
```

3. The Java SignalDemo Program

The program file `SignalDemo.java` contains the Java version of the above send/receive program. Read and understand the program; and then compile and run it.

Exercises

1. Comment our the line "`Thread.sleep(100);`" in the `main` function. Compile and re-run the program multiple times. Do you see different behaviors from these runs? Can you explain why?
2. Modify the original program to have two waiter threads, *e.g.*

   ```
   Thread waiter1 = new Thread(doWait, "Wait1");
   Thread waiter2 = new Thread(doWait, "Wait2");
   ```

   Run the program multiple times. Do you see different behaviors from these runs? Can you explain why?
3. Further modify the program, change the `synObj.notify()` call to:

   ```
   synObj.notifyAll();
   ```

   Run the new program. What happens?

4. The Pthread Barrier Program

Read and understand the program in file `barrier-pthd.c`.

Exercises

1. What do you think the program's output should be? Compile and run the program. Does it produce the expected result? Can you explain the program's output?
2. Insert barrier synchronization into the program, so that each of the statements in the `worker()` routine is guaranteed to have completed across all worker threads before the next statement starts. Compile and run the program. Does the output meets your expectation?

Submission

Write a short report, in plain text or pdf, summarize your experience with this lab and include your answers to the embedded questions. Zip your report and two program files, `condvar-pthd2.c` and the modified `barrier-pthd.c` into a single zip file, and submit it through the "Lab2" submission folder on D2L (under the "Activities/Assignments" tab). You should submit your work before the week-end, *i.e.* Sunday 1/20.