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 7: Programming with Chapel

Download and unzip the file `lab7.zip` from D2L. You'll see a `lab7` directory with some program files.

Use `addpkg` to add `chapel-1.12` to your environment.

Hello.chpl

Read, then compile and run the program `hello.chpl`. Try changing the message through the command-line a couple of times.

```
linux> chpl -o hello hello.chpl
linux> ./hello -nl 1
Hello, world!
linux> ./hello -nl 1 --message="Hi!"
Hi!
```

Note that to run a chapel program, you need to include the switch "`-nl 1`". This is because on our Linux system, the Chapel compiler is built to generate target code that is suitable for running across multiple locales. The switch "`-nl 1`" means running the program on a single locale.

Task-queue.chpl

The program `task-queue.chpl` contains the Chapel version of the task queue data structure that we used for Assignment 1. Read and understand the program. Then add the following statements to the `main()` routine to complete the program:

- Three statements for adding three new tasks to the queue.
- A `writeln` statement to display the content of the queue after the task insertions.
- Three statements for removing tasks from the queue.
- Three `writeln` statements, each after a task-removal statement, to show the queue's content.

Prodcons.chpl

The program `prodcons.chpl` is a Chapel version of the producer-consumer program. Read and understand the program. Pay special attention to the synchronization mechanism.

1. Why are some variables defined as synchronization variables, while others are not?
2. Compile and run this program.
3. Currently, the program creates only one consumer thread. Change the program so that multiple concurrent consumer threads are created. The number of consumer threads should be controlled by the parameter `numCons`.
4. (*Challenging!*) Currently, the program creates only one producer thread. Change the program so that multiple concurrent producer threads are created. The number of consumer threads should be controlled by a new parameter `numProds`.

Sum.chpl

The program `sum-pthd.c` is a Pthreads version of the sum program from Lab 1. Create a corresponding Chapel version.