CSc 422/522 — Homework 2

due Tuesday, February 20

Problems 1 to 4 are worth 5 points each. Problem 5 is worth 10 points. Problem 6 is worth 20 points. Again, graduate students are to solve all problems (50 points), and undergraduates are to solve any combination of problems that adds to 40 points.

Hand in written answers to problems 1 to 4, commented listings of your programs for problems 5 and 6, and sample output from your programs. *In addition*, use the `turnin` program to submit the source code for your programs; see the details at the end of the assignment. Be sure to explain (briefly) any assumptions you make when solving a problem.

1. MPD book, Exercise 3.3, parts (a) and (b).

2. MPD book, Exercise 3.3, part (c).

3. MPD book, Exercise 3.6.

4. MPD book, Exercise 3.8.

5. The purposes of this problem are to introduce you to the Pthreads library and to let you see the effects of not protecting critical sections. The Pthreads library is described in Section 4.6 of the text. You will also want to look at the manual pages for `pthreads` and for the various functions used in the four sample programs handed out in class. (These programs are stored in `/home/cs522/Programs/pthreads`.)

Compile your programs on Lectura but run your tests on Parallel so that processes execute concurrently. Parallel (par) has 6 processors; details are on the class Web page.

Assume that `x`, `y`, and `z` are *shared* integer variables that are all initially zero. Consider the following three statements:

```
S1:  x = x + 1;
S2:  y = y + 1;
S3:  z = z + x + y;
```

(a) Write a Pthreads program that has `numWorkers` processes. Each process executes the above three statements `numIters` times. Both `numWorkers` and `numIters` should be command-line arguments. At the end of the program, write out the final values of the three variables. Execute your program for several combinations of values of the command-line arguments. What do you observe? Why?

(b) Modify your program so that all three assignments execute as a single atomic action:

$\langle$`S1; S2; S3;`$\rangle$

Use Pthreads semaphores or mutexes (locks) to protect the critical sections in each process. Again execute your program for several combinations of values of the command-line arguments. What do you observe? Why?

6. MPD book, Exercise 3.28 or MPD book, Exercise 3.29.

Solve *either* of the above problems. You may write your program in MPD or in C plus Pthreads. Use semaphores to implement any atomic actions you need, but use busy waiting (spin loops) to program delays. Allocate space for all shared variables *before* you create the processes, and use arrays rather than linked lists.

Develop your programs on Lectura, but run your tests on Parallel. Turn in listings of your programs together with the output from the most interesting test runs.

If you choose to do the prime number generation problem, just implement the second approach described in Exercise 3.28. Your program should have two command-line arguments: `W`, the number of worker processes, and `L`, the largest number to check for primality. Seed the list of known primes with the first several odd primes. See how large a number you can check in a reasonable amount of time! To avoid contention for the bag, you may use separate bags for each worker process (but you will need to have a shared array of results).

If you choose to do the dictionary problem, read the dictionary into shared variables *before* you create the worker processes. The file `/usr/dict/words` contains 25,143 lines. Ignore the first few words, which begin with numbers, and ignore proper names, which begin with capital letters. Your program should have one command-line argument: `W`, the number of worker processes. It is up to you to figure out what to use for tasks, but a task should be *much* larger than checking a single word.

Electronic Turnin.

Use the `turnin` program on Lectura to turn in copies of your programs. For problem 5, the assignment name to use is `hw2.prob5`. The file names should be `parta.c` and `partb.c`.

For problem 6, the assignment name to use is `hw2.prob6`. The file name should be `primes.c`, `primes.mpd`, `words.c`, or `words.mpd`.