CSc 422 — Homework 1

Due Tuesday, February 8, 2005

This assignment is worth 40 points, divided as indicated. Turn in written answers to the first five problems. For the programming assignments, turn in nicely-commented listings *as well as* answers to the questions about each program that I ask below. In addition, submit your programs electronically. See the end of this assignment for information on programming style and electronic turnin.

You may discuss the meanings of questions with classmates, but the answers and programs that you turn in must be yours alone. For the exercises from the book, explain your answers clearly and succinctly.

1. [6 points] MPD book, Exercise 2.3

2. [6 points] MPD book, Exercise 2.8

3. [4 points] MPD book, Exercise 2.10

4. [2 points] MPD book, Exercise 2.19

5. [2 points] MPD book, Exercise 2.25

6. [10 points] **Atomic vs. Nonatomic Execution.** The purpose of this problem is to let you see the effects of not protecting critical sections of code. Write your programs either in C with the Pthreads library or in the MPD language (or in both!). Compile your programs on Lectura (`lec`), but run your tests on Parallel (`par`) so that threads execute concurrently. Parallel has 6 processors; details are on the class Web page.

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

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

(a) Write a program that has `numWorkers` processes (threads). Each process executes the above three statements `numIters` times. Both `numWorkers` and `numIters` should be command-line arguments, in that order. At the end of the program, write out the final values of the three variables.

Execute your program for one, two, and three workers, and for 1000, 5000, and 10000 iterations. This is a total of nine different test runs. What do you observe? Why? (If you use MPD, be sure to set the `MPD_PARALLEL` environment variable each time you change the number of workers.)

(b) Modify your program to use two atomic actions, as follows:

```
⟨S1; S2;⟩
⟨S3;⟩
```

Use a semaphore or a mutex lock to protect the two critical sections in each process. (They are not disjoint, so you need to protect them with the same semaphore or lock to make them appear to be atomic.) Repeat the same set of experiments you used for part (a), and answer the same questions.

7. [10 points] **Parallel Processing**. The purpose of this exercise is to introduce you to parallel programming. Given is a matrix of NxN integers. The matrix is initialized to random values between 1 and 10 inclusive. The problem is to calculate the distribution of values, i.e., the number of elements that are 1, the number that are 2, etc.

Divide the work up among W worker processes (threads). The matrix is shared by the workers. You may assume that N is a multiple of W. Divide the matrix into W equal-size strips of N/W rows. Each worker process should fi rst initialize all the elements in its strip of the matrix to random values between 1 and 10, and then calculate the distribution of values for its strip. When all workers have completed their tasks, one of the workers should calculate the fi nal answer and print the results. Use flag variables to synchronize the printing of results. Be careful to avoid read/write interference between the workers as they calculate results for their strips.

Write a Pthreads or MPD program to solve the above problem. Use the C or MPD random number function to generate psuedo-random numbers. Again, develop your programs on Lectura, but run your experiments on Parallel. The values of N and W should be read as command-line arguments. Write the results to standard output.

Run experiments for N equals 64, 128, and 256 and W equals 1, 2, and 4. This is again a total of nine test runs. Briefly describe your results. What do you observe? Why?

Programming Style. Your programs should be easy to read and self-contained. At a minimum you should:

(1) Include a descriptive header comment with your name, a brief description of the program, and the command-line arguments.

(2) Give short but descriptive comments for each variable, process, procedure, and signifi cant block of code.

(3) Use a *reasonable* level of indentation but not too much. I think 3 or 4 spaces is plenty; 8 is usually too many. Your printed listing should not have huge amounts of white space and should not have long lines that get wrapped around when printed.

Be sure to include your name in the header comment.

Electronic Turnin. Use the `turnin` program on Lectura to submit electronic versions of your program so that we can run some tests. See the man page for `turnin` for details.

For problem 6, the assignment name is `hw1.prog1`. The fi le names should be `nolocks.c` and `locks.c` or `nolocks.mpd` and `locks.mpd` for parts (a) and (b), respectively.

For problem 7, the assignment name is `hw1.prog2`. The fi le name should be `random.c` or `random.mpd`.