

CSc 422 — Homework 1

Due Tuesday, February 10, 2009

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 and sample output, *as well as* answers to the questions about each program that I ask below. In addition, submit the source code for 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. [4 points] MPD book, Exercise 2.5
2. [4 points] MPD book, Exercise 2.10
3. [4 points] MPD book, Exercise 2.13
4. [4 points] MPD book, Exercise 2.15
5. [4 points] MPD book, Exercise 2.25
6. [10 points] **Atomic vs. Nonatomic Execution.** The purpose of this problem is to demonstrate 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 Voltron (`vol`) so that threads execute concurrently. Both machines have 4 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] **Concurrent Grep.** In class, and in Section 2.2 of the text, we discussed the problem of finding a pattern in a file. Here you are to look for a pattern in one or more files. If the name of your program is `cgrep`, the command line arguments are:

```
cgrep pattern file1 file2 ... fileN
```

Write your programs using MPD or C plus Pthreads. Again, develop your programs on Lectora, but run your experiments on Voltron. We will provide some test data that you can use.

(a) Write a program that reads the command line arguments and then forks one process to handle each file. Each process should read every line of the appropriate file, and if the pattern is found in that line, write the line to `stdout`. The output from different processes might get interspersed, but that is OK in this program.

(b) Now modify your program so that the output appears in the same order as the list of files. That is, the first process (which reads `file1`) should print all its output before the second process, which should precede the third process, and so on. To enable each process to read and examine its file in parallel with the other processes, use a local array of strings to buffer the output until it is time to write it to `stdout`. Use either signaling flags or semaphores to synchronize the processes.

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 significant 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 Lectora 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.prob6`. The file 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.prob7`. The file names should be `cgrep.unordered.c` and `cgrep.ordered.c` or `cgrep.unordered.mpd` and `cgrep.ordered.mpd` for parts (a) and (b), respectively.