

CMSI 387

OPERATING SYSTEMS

Spring 2008

Assignment[s] 0424

To reconcile the amount of time remaining in the semester (plus your kernel project) against the amount of exercise/reinforcement that I think you should get with the remaining material, this assignment bears some differences from previous ones:

- Each segment of the assignment actually counts as an assignment in and of itself, so you may submit only as much as you feel the assignment part of your grade needs (or, do the easiest jobs first — “EJF”). Of course, accomplishing the harder stuff comes with extra-credit rewards.
- This assignment gives you a little more time (an additional week) while still allowing me to review, correct, and discuss them before the final.
- The assignment already includes work for material that has not yet been covered, in case you’d like to read ahead and get a jump on them before we get to them in class.

We had to make some compromises here, but I hope these tweaks still make the assignments worth doing while allowing enough time to do them while working with your other end-of-semester deliverables.

Not for Submission

We have covered SGG Chapters 3, 4, and 5 so far, and are in the middle of Chapter 6. The plan is to make it to Chapter 9 by the end of the semester.

For Submission

A Shell of Your Own

Modify either the *fork-exec.c* program given out in class or the starter code that is referenced at the end of SGG Chapter 3 to implement your own command-line operating system shell. Commit your code to `/homework/cmsi387/myshell`.

Your shell must, at a minimum, implement the basic command-prompt loop that is typical of most shells, as well as an “&” directive that tells the shell *not* to wait for an invoked command to finish. You will get extra credit if you also implement the command history feature mentioned at the end of SGG Chapter 3.

Concurrent Matrix Multiplication

Do the project at the end of SGG Chapter 4, which implements matrix multiplication using threads. Use Pthreads for this assignment, and commit your code to `/homework/cmsi387/matrix`. Note that, for those in CMSI 371, you should already have a Java-based, single-threaded version of this code already done, so this assignment should just feel “incremental” to that one.

Dining Philosophers

Implement a solution to the dining philosophers problem using the POSIX API. The bounded buffer code [to be] given out in class may be used as a basis for your solution.

Make sure to include well-placed output statements to report what’s happening in your program and the state of things at any given time — that’s how we’ll know whether your solution is working. Commit your code to `/homework/cmsi387/dp`.

You will get extra credit if you also implement a solution to the sleeping barber problem (SGG Exercise 6.11). Commit this to `/homework/cmsi387/sb`.

Exercises

Do the following exercises from SGG; submit your answers in hardcopy.

1. SGG Exercises 6.1 and 6.17 (*Hint*: The answer to 6.17 can be found in one of the “classic papers” given out in class.)
2. Name two things (for a total of four observations) that may happen in *incorrect* critical section solutions to the dining philosophers and sleeping barber problems.
3. SGG Exercises 7.2 and 7.11
4. SGG Exercises 8.3, 8.6, and 8.9
5. SGG Exercises 9.3, 9.5, and 9.14