CSCI 3323 (Principles of Operating Systems), Fall 2013

Homework 3

Credit: 10 points.

1 Reading

Be sure you have read Chapter 2, sections 2.1 through 2.3 and 2.5

2 Problems

Answer the following questions. You may write out your answers by hand or using a word processor or other program, but please submit hard copy, either in class or in my mailbox in the department office.

1. (5 points) Solve the dining philosophers problem with monitors rather than semaphores. (Do this yourself, though, rather than looking for a solution online or in another book!)

2. (5 points) Restrooms are usually designated as men-only or women-only, but this requires having two restrooms if everyone is to be accommodated. A less expensive approach consistent with cultural norms in the U.S. would be to have one restroom with a sign on the door that indicates its current state — empty, in use by at least one woman, or in use by at least one man. If it is empty, either a man or a women may enter; if it is occupied, a person of the same sex may enter, but a person of the opposite sex must wait until it is empty. Write pseudocode for four functions to implement this approach: `woman_enter`, `man_enter`, `woman_leave`, and `man_leave`, to be used by the following pseudocode:

```
/* woman process */
while (TRUE) {
    woman_enter();
    use_restroom();
    woman_leave();
    do_other_stuff();
}
/* man process */
while (TRUE) {
    man_enter();
    use_restroom();
    man_leave();
    do_other_stuff();
}
```

   You can use any of the synchronization mechanisms we have talked about (shared variables, semaphores, monitors, or even message passing).

3 Programming Problems (Optional)

Do the following programming problems. You will end up with at least one code file per problem. Submit your program source (and any other needed files) by sending mail to `bmassing@cs.trinity.edu`, with each file as an attachment. Please use a subject line that mentions the course number and the assignment (e.g., "csci 3323 homework 3"). You can develop your programs on any system that provides the needed functionality, but I will test them on one of the department's Linux machines, so you should probably make sure they work in that environment before turning them in.

1. (Optional — up to 5 extra-credit points) Write a program to test your solution to either problem 1 or problem 2. If you want to do this using C and POSIX threads, you could start with the code for the programming problem in Homework 2. Or you could write in Java and use its monitor-based synchronization (synchronized methods/blocks plus `wait`, `notify`, and `notifyAll`) and/or features of the `java.util.concurrent` library package (which has, among many other things, a `Semaphore` library class). You can find some simple examples of multithreaded Java programs on the "Sample programs" page for my parallel programming class: http://www.cs.trinity.edu/ bmassing/CS3366/SamplePrograms/[1]. The bounded buffer example may be useful if you want to use monitor-based synchronization.

[1] `http://www.cs.trinity.edu/~bmassing/CS3366/SamplePrograms/`