CSCI 3323 (Principles of Operating Systems), Fall 2021

Homework 4a

Credit: 15 points.

1 Reading

Be sure you have read, or at least skimmed, Chapters 25, 26, 27, 28, 30, 31, 32, and 34 of the textbook. (Yes, I'm skipping chapters 29 and 33.)

2 Problems

Answer the following questions. You may write out your answers by hand and scan them, or you may use a word processor or other program, but please turn in a PDF or plain text file. (No links to shared files on Google Drive please, and no word-processor files.) Turn it in by putting it in your course "TurnIn" folder on Google Drive. Please be sure to include your name somewhere in the file, so when I print it for grading I know whose work it is. (In the pledge is fine.)

1. (7.5 points) Consider a simple print-spooling system in which all processes that want to print have access to a shared "print queue" (a list of filenames to be printed), and one process actually performs the printing. Pseudocode for the two kinds of processes (those putting things in the print queue, and the one process actually printing them) might look like the following. Variable `printQueue` is shared among processes, but all other variables are local to a method/process.

   Printer process:

   ```
   while (true) {
       outFileName = generateFileToPrint();
       enqueue(printQueue, outFileName);
   }
   ```

   Printer user process:

   ```
   while (true) {
       if (!empty(printQueue)) {
           fileToPrint = dequeue(printQueue);
           print(fileToPrint);
       }
   }
   ```

   Do we need some sort of locking to make this work right? If so, say how many locks you think would be needed and where you would put calls to `lock()` and `unlock()`.

2. (7.5 points) Consider again our semaphore-based solution to the mutual-exclusion problem:

```
// shared variables
semaphore mutex(1); // initial value 1

// process
while (true) {
    down(mutex);
    do_critical_region();
    up(mutex);
    do_non_critical_region();
}
```

Answer the following questions.

(a) Would this solution still work (i.e., guarantee that only one process at a time is in its critical region, no process waits forever, etc.) if the initial value of the semaphore `mutex` were 0? Why or why not?

(b) Would this solution still work if the initial value of the semaphore `mutex` were 2? Why or why not?

3 Essay and pledge

Include with your assignment the following information.

For programming assignments, please put it a separate file. (I strongly prefer plain text, but if you insist you can put it in a PDF — just no word-processor documents or Google Drive links please.) For written assignments, please put it in your main document.

3.1 Pledge

This should include the Honor Code pledge, or just the word "pledged", *plus* at least one of the following about collaboration and help (as many as apply). Text *in italics* is explanatory or something for you to fill in; you don't need to repeat it!

- I did not get outside help *aside from course materials, including starter code, readings, sample programs, the instructor.*
- I worked with *names of other students* on this assignment.
- I got help with this assignment from *source of help — ACM tutoring, another student in the course, etc. (Here, "help" means significant help, beyond a little assistance with tools or compiler errors.)*
- I got help from *outside source — a book other than the textbook (give title and author), a Web site (give its URL), etc.. (Here too, you only need to mention significant help — you don't need to tell me that you looked up an error message on the Web, but if you found an algorithm or a code sketch, tell me about that.)*
- I provided help to *names of students* on this assignment. *(And here too, you only need to tell me about significant help.)*

3.2 Essay

This should be a brief essay (a sentence or two is fine, though you can write as much as you like) telling me what if anything you think you learned from the assignment, and what if anything you found interesting, difficult, or otherwise noteworthy.