CSCI 3323 (Principles of Operating Systems), Fall 2020

Reading Quiz 3

Credit: 10 points.

1 Reading

Be sure you have read, or at least skimmed, sections 2.3 and 2.5 of Chapter 2.

2 Instructions

Answer the questions below using *only* the course textbook (i.e., no Web searches). Please work independently rather than in groups, and include the Honor Code pledge in what you turn in, either the full pledge or just the word "pledged".

You may write out your answers by hand and scan them, or you may use a word processor or other program, but please submit a PDF or plain text via e-mail to my TMail address. (No links to shared files on Google Drive please.) Please use a subject line that mentions the course and the assignment (e.g., "csci 3323 quiz 3" or "O/S quiz 3").

3 Questions

1. (2 points) The textbook's discussion of implementing semaphores includes the same `enter_cr()` assembly-language function as the one shown in class, except that the textbook discussion talks about using pure busy-waiting rather than what we did in class, which is to allow another process to run if the function doesn't acquire the lock. Given that part of the point of semaphores is to avoid busy-waiting, why isn't this a serious problem?

2. (2 points) Why are semaphores easier to implement than monitors?

3. (2 points) In what situations is not possible to use synchronization mechanisms based on shared memory, and what could you use instead?

4. (2 points) The discussion of message-passing in the textbook describes one difficulty in implementating this mechanism, namely the fact that networks are not always 100% reliable, and messages can get lost. Another potential problem is that messages from process A to process B might not arrive in the order in which they were sent. Many specifications for message-passing systems require that messages arrive (in the sense of being available to receive) in the order sent. What might be a way to guarantee this, if the technique described for coping with faulty networks is being used?

5. (2 points) In the textbook's discussion of the dining philosophers problem, the authors first explain that the first-thought "solution"' of just having a hungry philosopher pick up its left fork and then its right fork can deadlock. It goes on to say that it's tempting to fix this by having philophers first check whether the right fork is available and if not put down the left fork, delay, and then try again. What are two problems with this approach?