Synchronization

Wednesday, October 28, 2015 11:40 AM

Synchronization Worksheet

You no not need to turn-in this sheet. Its purpose is to help you identify relevant ideas presented during the lecture and to review topics.
(In-class team discussion problems are located in the OneNote--> Discussions)

1. What is a race condition?
   Make sure that you can create (or recall) an example that illustrates how a race condition can cause problems in your program.

2. What are our three requirements for code that executes a critical section?
   For each one, make sure that you know the 'formal' name listed in the slides, and that you're able to clearly explain what it means ***in your own words***.

3. Explain how the 'strict turn-taking' approach attempted to limit access to a critical section, and also explain why it fails to satisfy the criteria listed in the prior question (be which about which one it violates and why).
   You should explain this conceptually/intuitively (i.e., using clear English, NOT using code)

4. Answer the same question as #3, but this time for the 'one flag per process' approach

5. Explain how Peterson's solution works. Including code here would be good.
6. Also: Make sure to clearly identify how many processes may execute Peterson's algorithm!

7. Explain why/how Peterson's solution meets all three criteria.

8. Why is it important to keep critical sections as small as possible?

9. What is interrupt masking used for, and how does it work? What are some drawbacks of using this approach?

10. What is a "test-and-set" instruction? What is "compare and swap"? How do they differ from 'normal' instructions?

11. Explain (conceptually, intuitively) how we can use a "test-and-set" instruction in order to protect a critical section.

12. Explain how we can use an "atomic swap" instruction in order to protect a critical section.
    Please feel free to do this by citing your prior explanation and then explaining how the swap instruction can be used in place of the test-and-set (i.e., explain this by focusing on how you can substitute the "swap" instruction into more-or-less the same logic as you used in the prior question).

13. What is a semaphore? How does it relate to the "test-and-set" material that was covered prior to this?