

ITEE



What's New:

There are fewer questions this time, since some of you may want to focus on your assignment. Some of the questions are more conceptual than before, and may take some thinking through.

Learning objectives for this week:

- Understand the major concepts we've covered, including:
 - synchronization
 - hardware vs. software primitives
 - classic problems and solutions
- Apply these concepts to reasoning about synchronization primitives and algorithms using them
- Apply these concepts to choosing an appropriate primitive in a given situation
- Understand where and how deadlock and starvation occur

As before, you should aim to answer all of these questions by the end of the course. Being able to answer them by the end of week 6 is a useful goal. Be prepared: if you know what you can't do yourself, you will make most efficient use of the tutorial.

1. Concepts

- a. Why is it useful to have hardware primitives to support synchronization?
- b. Why is it not desirable to have race conditions?
- c. Explain why high-level primitives like monitors are useful.

2. Low-Level Primitives

- a. Try to find an example of use of the Bakery Algorithm (pp 196-197) which convinces you that it works (or doesn't).
- b. Why is a spinlock (Figures 7.7-8, pp 198-199) not suitable as a general synchronization mechanism for a single-processor CPU? When would you use a spinlock?
- c. Work through an example using both the pseudocode semaphore definition (p 201) and the "real" algorithm (p 203) with the initial count of semaphore $s = -2$, and 3 processes defined as follows, assuming the processes run in order P_0, P_1, P_2, P_3 , i.e., P_3 reaches its signal after the others have already executed their wait:

```
P0:
    wait (S)
    signal (S)
```

```
P1:
    wait (S)
    signal (S)
```

```
P2:
    wait (S)
    signal (S)
```

```
P3:
    signal (S)
```

- i. Is there any difference between what the two versions of the semaphore algorithm do *in this example*?
- ii. For both versions of the semaphore algorithm, explain what a negative initial value for s signifies with this specific use of semaphores.
- iii. Are the two versions of the semaphore algorithm the same *in general*? If not, list the differences. If so, justify your answer.

3. Classic Problems

- a. Bounded buffer (algorithm p 207).
 - i. Show that if the buffer is full, a producer will be blocked, but a consumer will not be.
 - ii. Show that if the buffer is empty, a consumer will be blocked, but a producer will not be.
 - iii. Show that if the buffer is neither full nor empty, neither the producer nor the consumer will be blocked *unless the other is in its critical section* (between `wait(mutex)` and `signal(mutex)`).
 - iv. In view of (i)–(iii), is the given bounded buffer generally correct?
- b. Dining Philosophers.
 - i. The simplest approach (p 220) can easily lead to deadlock as described in the book. Show a sequence of operations which will lead to deadlock.
 - ii. Show that the solution using a monitor (p 219) can lead to starvation.

4. Work through any problems you may have arising out of week 5 lectures.