

1. What is the difference between semaphores and condition variables?

Semaphores store 'history'—that is, if you increment a semaphore and no thread is waiting on it, the next thread to come along and check that semaphore will see that its greater than 0 and continue. Condition variables don't have history. If you signal() and no thread is waiting, that signal is lost.

2. A simple implementation of a lock is given below. Why must the interrupts be disabled in the **Release** procedure?

```
class Lock {
    int value = FREE;

    Acquire() {
        Disable interrupts
        if (value == BUSY) {
            put on wait queue, sleep, disable
interrupts
        } else {
            value = BUSY;
        }
        Enable interrupts;
    }

    Release() {
        Disable interrupts;
        if (wait queue not empty) {
            take thread off wait queue, add to
ready queue
        } else {
            value = FREE;
        }
        Enable interrupts;
    }
}
```

If the release method did not disable interrupts, a thread could check to see if a thread needs to be taken off the wait queue and then be context switched out of the processor before releasing the lock. Another thread attempts to acquire the lock, finds that it is not available, adds itself to the waiting queue, and sleeps. When the initial thread wakes and releases the lock, there is a thread sleeping on the waiting queue with no way to wake it up

3. Suppose we had the following situation:

- There is a box of donuts on the counter.
- Jim goes to the kitchen to eat a donut.
- Fred also goes to the kitchen to eat a donut.
- Bob the health nut doesn't eat donuts, but instead puts a single donut into the donut box every so often for his roomies to eat.

How would you write the code for the threads corresponding to Jim, Fred, and Bob so that each of them can fulfill their tasks? Hint: you might find the synchronization types we talked about in class helpful!

Here's one solution that involves a monitor:

Condition d;

Lock on donut box;

Jim and Fred:

Acquire lock on donut box

Wait on d

Take a donut out of the box

Release the lock

Bob:

Acquire lock on donut box

Put a donut in the box

Signal on d

Release donut box lock.