CS377 Operating Systems
Quiz 1
SOLUTIONS

April 7, 2003

Q1. List two key distinctions between a semaphore and a monitor.

1. A monitor implicitly enforces a sequence of acquire/release operations; a semaphore does not.
2. The monitor notify() does not have a memory, whereas signal() does. In other words, when there are no waiting threads a monitor notify() is equivalent to a no-op.
3. A monitor may sleep inside of its "critical section." In doing so, it implicitly releases the mutex, thus allowing other threads to enter.
4. A monitor wraps a data structure within the synchronization, enforcing proper use. Semaphores rely completely on the programmer for proper use.

Q2. What is the distinction between deadlock and starvation?

Deadlock is the situation in which a process is waiting on another process, which is also waiting on another process ... which is waiting on the first

process. None of the processes involved in this circular wait are making progress.

With starvation, a process is waiting (possibly by recursion) on a process or set of processes that are (possibly!) making progress.

Several people gave answers talking about a starved process not getting the CPU (referring to our scheduling discussions). More generally, starvation is about a process not getting the resources that it needs to make progress (of which CPU is but one type of resource).

Several people also talked about deadlock in terms of "all processes." In the general case, only a subset of processes may be involved in the deadlock while other processes (not involved in the waiting relationship) are making progress.

Note that deadlock implies that the processes involved in the deadlock are starved. But - starvation does not imply that deadlock has occurred.