Homework 1 (due 9/18/2008 at 12:05pm)
via email to ada@cc or hardcopy in class

1. Regular mutexes block when they are already locked and a thread A tries to lock them, even if the current holder is the same thread. In contrast, a *recursive mutex* is one that can be locked multiple times by the same thread and needs to be unlocked the same number of times before other threads can enter the critical section. Using the primitives shown in class, or any equivalent set you are familiar with, implement a recursive mutex. That is, with pseudo code (or pthreads if you want to) show the implementation of:

- a type `RecMutex`
- three routines `rec_mutex_lock`, `rec_mutex_unlock`, and `rec_mutex_init` that operate on entities of type `RecMutex` and implement the entry and exit from the critical section protected by a recusive mutex, as well as the initialization of the recursive mutex data.

You may assume a function `CurrentThread()`, returning a unique identifier for the currently executed thread.

2. Consider a system running ten I/O-bound tasks and one CPU-bound task. Assume that the I/O-bound tasks issue an I/O operation once for every millisecond of CPU computing and that each I/O operation takes 10 milliseconds to complete. Also assume that the context switching overhead is 0.1millisecond and that all processes are long-running tasks.
a) What is the CPU utilization for a round-robin scheduler when the time quantum is 1 millisecond?
b) What is the CPU utilization for a round-robin scheduler when the time quantum is 10 milliseconds?
c) Explain how may the following scheduling criteria conflict in certain cases: I/O device utilization and CPU utilization.