Homework 1 (due 1/30/2006 at 10:05am)

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, implement:

- 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. This question has to be answered with respect to Solaris. Explain the performance impact of context switching from a thread T1 to a thread T2 in each of the following situations:

a) T1 and T2 are both user level threads within the same process and are attached to the same LWP.
b) T1 and T2 are both user level threads within the same process but attached to different LWPs.
c) T1 and T2 are user level threads in different processes.
d) T1 and T2 are kernel threads unattached to any user level threads.

In each case, you should clearly identify the source of any performance hit that the thread switch would entail

3. *Optional, interview-style "tricky" question, just for fun:* For regular (non-recursive) mutexes, we have used a block-structured "Lock" construct (i.e., unlocking a mutex is implicit at the end of the statement/block under the "Lock"). Can you use C macros to define a block-structured "Lock" construct from a pair of lock/unlock routines? (C++-style local variable definitions are ok, in case you need them. Alternatively, you may need to change the lock/unlock routines slightly to control what they return.)