CSE 451 Homework Assignment #2

Out: April 6, 2000
Due: April 12, 2000

#1. When a physical device needs attention from the system it generates an interrupt. An interrupt in this case is handled like a mini-context switch. It only saves those registers used by the interrupt handling code rather than swapping out the entire register set as would occur in a normal context switch. Think about why this is true. List and explain briefly one advantage and one limitation of using mini-context switches over full-blown context switches to handle hardware interrupts.

#2. Describe the similarities and differences of doing a context switch between two processes as compared to doing a context switch between two threads in the same process.

#3. If you were given the task of writing a program using either multiple processes, each single-threaded, or a single process with multiple threads, what would be three important design considerations you should take into account when making your decision?

#4. This question requires you to use the nachos code provided on the web page. The instructions for setting up nachos is on the course web page. (These will be up by Thursday night or Friday morning.) To gain some initial experience with nachos, it is good to understand how threads are implemented in this operating system, so this problem will require you to perform a simple trace of what happens when a thread yields the CPU to a newly created thread.

• There is a newly created thread (that has never been run) on the ready list in the scheduler.
• There is a thread currently running that executes kernel->currentThread->Yield();

Trace through the code and explain what happens on the Yield call. Make sure to include information about when interrupts are enabled and disabled and what exactly happens on the context switch. It is very important to understand what happens in switch.s with the register swaps and what happens when we “return” to the new thread.

You only have to pay attention to calls and returns to routines defined in thread.cc, scheduler.cc, and switch.s, and enabling/disabling of interrupts. In other words, you can ignore ASSERT statements, DEBUG statements, calls to list routines, calls made internal to the interrupt simulation, and any trivial inline functions. You should also assume that timer interrupts are disabled, so there is no preemption.

If it will help you trace, feel free to run a sample program and print out a trace, but make sure to explain what is going on at key points in the program.

When doing this problem, you may stop your trace once the new thread is actually running the code designated by “InitialPC” in switch.s.