HW 2-19 6 Problems due Friday

1. Draw with pencil or graphics software two neat sequence diagrams to illustrate each history:

   (a) One process makes a non-blocking system call to the operating system. The operating system returns directly to that process after handling the system call.

   (b) One process makes a blocking system call. The OS kernel runs a second (ready) process. This second process is interrupted by the event caused by the I/O device where this event enables the blocked system call to complete. (You must show a column for the I/O device and indicate when the I/O device sends an interrupt that interrupts the second process.) The OS kernel reschedules the interrupted 2nd process because it has high priority. After a timer interrupt, the OS kernel makes the second process be idle (in the ready state) and makes the previously blocked system call return to the original process.

2. (Tanenbaum, Ch. 2, Prob. 3) On all current computers, at least part of the interrupt handlers are written in assembly language. Why? List and briefly describe as many different operations of the interrupt handler and subsequent kernel code that cannot be written in pure C.

3. (Tanenbaum, Ch. 2, Prob. 8)

4. (Tanenbaum, Ch. 2, Prob. 10)

5. (Tanenbaum, Ch. 2, Probs. 23 and 24) Include short justifications for each answer. What if the scheduling is non-preemptive and there is a call to yield() inside the loop of Peterson’s algorithm?

6. Discuss in as much depth as needed the importance of choosing the right value for the quantum when the scheduler is non-preemptive.

HW 2-24 3 to 6 Problems Due Wed, some programming

1. (Tanenbaum, Ch. 2, Prob. 37)

2. (Tanenbaum, Ch. 2, Probs. 40 and 41)

3. (Tanenbaum, Ch. 2, Prob. 51) Create, debug, test and analyze with programs using pthreads. Two versions are required: One should use whatever counters and synchronization techniques you like to achieve a solution. The other should use the strategy similar to your correct solution but omits mutual exclusion. Include in your submission a single large enough test case to demonstrate failure of the second program and success of the first. (It might be hard-coded into the driver program.) Write a description of your testing that documents how well the demonstrations succeeded.

4. **Required for grad. students only.** (Tanenbaum, Ch. 2, Prob. 35)

5. **Required for grad. students only.** (Tanenbaum, Ch. 2, Prob. 47)

6. **Required for grad. students only.** (Tanenbaum, Ch. 2, Prob. 49) Actually implement using pthreads semaphores, write a solution to the producers-consumers problem using your message passing solution, and do stress-testing of it.