CSCI 4320 (Principles of Operating Systems), Fall 2002

Homework 2

Assigned: October 3, 2002.

Due: October 10, 2002, by 5pm.

Credit: 30 points.

1 Reading

Be sure you have read chapter 2.

2 Problems

Answer the following questions. You may write out your answers by hand or using a word processor or other program, but please submit hard copy, either in class or in my mailbox in the department office.

1. (5 points) Does a timesharing system need a process table? Why or why not? What about a personal-computer system in which only one process at a time can execute, that process taking over the whole machine until it is finished? Why or why not?

2. (5 points) Look again at the solution to the mutual-exclusion problem presented in Figure 2-20 in the textbook. If the two processes are running on a computer with two CPUs and a common memory, does this solution work? I.e., which if any of the criteria given on p. 102 does it satisfy? Briefly justify your answer.

3. (5 points) Consider a computer that does not have a test-and-set-lock (TSL) instruction, but does have an instruction to swap the contents of a register and a memory word in a single indivisible action. Use such an instruction (call it SWAP) to write a routine *enter_region* like the one found in Figure 2-22 in the textbook, or explain why this is impossible.

4. (5 points) Give a sketch (possibly pseudocode) of how you could implement semaphores on a single-CPU system on which the operating system can disable interrupts.

5. (5 points) In the solution to the dining philosophers problem shown in Figure 2-33 in the textbook, why is the state variable set to *HUNGRY* in the procedure *take_forks*?

6. (5 points) Consider the procedure *put_forks* in Figure 2-33 in the textbook. Suppose that the variable *state[i]* was set to *THINKING after* the two calls to *test* rather than before. How would this change affect the solution? (I.e., would it work as well as before? better? not as well?)