1997-1998

Winter 1998 Comprehensive Exam: Software Systems (30 points total)

1. (15 points) Some possible criteria for a process scheduler are the following:

   1. Fairness: make sure each process gets its fair share of the CPU.

   2. Efficiency: keep the CPU busy with useful work as much as possible.

   3. Response time: minimize response time for interactive users.

   4. Turnaround: minimize the time batch users must wait for output.

   5. Throughput: maximize the number of jobs processed per hour.

   Please answer the following questions given these criteria.

   a. Which of these criteria are contradictory with each other, and why?

   b. In a system with only small interactive processes, all of which are of equal priority, what would be a good scheduling algorithm? Why? (Pick as simple an algorithm as you think would be suitable.)

   c. Let's say we add processing and display of a real-time video feed to this system otherwise dominated by small interactive processes. What kinds of new scheduling problems or issues arise? (You do not need to come up with a scheduling algorithm to solve them.)

   d. Consider a system with more than just small interactive processes. Besides the interactivity, priority and real-time nature of a process, what other characteristics of a process might you take into account in the scheduler?

   e. What technology trends would have an effect on the importance of the characteristics you listed in question d?

2. (5 points) What is a race condition? Give an explicit example.

3. (10 points) Upon rebooting after a crash, a boot program checks for the consistency of the file system. Let's assume the system uses a standard UNIX file system. The program first checks for file block consistency. To do so, it builds a table with two counters per block, both initially 0. The first counter keeps track of how many times the block is present in a file; the second records how often it is present in the free list (or bit map of free blocks). The program then reads all the i-nodes. Starting from an i-node, it is possible to build a list of all the block numbers used in the corresponding file. As each block number is read, its counter in the first table is incremented. The program then examines the free list or bit map, to find all the blocks that are not in use. Each occurrence of a block in the free list results in its counter in the second table being incremented. (The system also checks for consistency of the directory structure, but you can ignore that.)

   a. What are all the types of inconsistencies that could be found, and how does the program detect them from these two tables? (One of these inconsistencies is impossible if the system uses a bit map rather than free list.)

   b. For each of these inconsistencies, how would the program fix the problem?