CSCI 4320 (Principles of Operating Systems), Fall 2006

Homework 4

Assigned: November 17, 2006.

Due: November 29, 2006, at 5pm.

Credit: 30 points.

1 Reading

Be sure you have read (or at least skimmed) Chapter 5.

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) The textbook divides the many routines that make up an operating system's I/O software into four layers, as shown in Figure 5-10. In which of these layers should each of the following be done? Why? (Assume that in general functionality should be provided at the highest level at which it makes sense — e.g., in user-level software rather than device-independent software.)

   (a) Converting floating-point numbers to ASCII for printing.

   (b) Computing the track, sector, and head for a disk read operation.

   (c) Writing commands to a printer controller's device registers.

   (d) Detecting that an application program is attempting to write data from an invalid buffer address. (Assume that detecting an invalid buffer address can only be done in supervisor mode.)

2. (5 points) Consider a printer that prints at a maximum rate of 400 characters per second, connected to a computer system in which writing to the printer's output register takes essentially no time. If each character printed requires an interrupt that takes a total of 50 microseconds (i.e., 50×10^{-6} seconds) to process, would it make sense to use interrupt-driven I/O to write to this printer, or would it be better to use programmed I/O? Why? (*Hint:* How much time is required for interrupt processing if the printer is printing at its maximum rate?)

3. (5 points) Now consider a system with a memory-mapped terminal, and suppose that interrupts take a minimum of 100 nsec to process and copying a byte into the terminal's video RAM takes 10 nsec. Would it make sense to use interrupt-driver I/O to write to the terminal, or would it be better to use programmed I/O? Why?

4. (5 points) Consider a computer system that maintains date and time using a 32-bit unsigned integer whose value represents a number of seconds since January 1, 1970. (So, a value of 362 would represent 12:06:02 am, January 1, 1970.) In what year will this scheme become unworkable because the 32-bit integer is not big enough? What if instead the system uses a signed 32-bit integer, allowing negative values to represent dates and times before January 1, 1970? (Ignore leap-year complications and assume that the average year has 365.25 days.)

5. (5 points) Suppose at a given point in time a disk driver has in its queue requests to read cylinders 10, 22, 20, 2, 40, 6, and 38, received in that order. If a seek takes 5 milliseconds (i.e., 5×10^{-3} seconds) per cylinder moved, and the arm is initially at cylinder 20, how much seek time is needed to process these requests using each of the three scheduling algorithms discussed (FCFS, SSF, and elevator)? Assume that no other requests arrive while these are being processed and that for the elevator algorithm the initial direction of movement is outward (toward larger cylinder numbers).

6. (5 points) Student H. Hacker installs a new disk driver that its author claims improves performance by using the elevator algorithm and also processing requests for multiple sectors within a cylinder in sector order. Hacker, very impressed with this claim, writes a program to test the new driver's performance by reading 10,000 blocks spread randomly across the disk. The observed performance, however, is no better than what would be expected if the driver used a first-come first-served algorithm. Why? What would be a better test of whether the new driver is faster? (*Hint:* The test program reads the blocks one at a time. Think about how many requests will be on the disk driver's queue at any one time.)