CPSC 431 - Operating Systems

Instructor Marc Corliss

Homework 2 Solutions

Due: 9/23/09 (beginning of class)

1. Describe in some detail (at least 3 sentences) the following:

   (a) Control registers
   Answer:
   Control registers are used for programming devices. The CPU has instructions for reading and writing special registers, which are visible to various devices. Each control register is assigned an I/O port number and an 8-bit or 16-bit integer, which specify what device to read or write.

   (b) Memory-mapped I/O
   Answer:
   Memory-mapped I/O maps control registers into the memory space. Particular frames are dedicated for I/O devices. Writing to these frames is observed by the corresponding I/O device controller, which handles the request.

   (c) DMA
   Answer:
   Direct memory access uses additional hardware to perform copies between memory and I/O devices. This frees the CPU to perform other tasks, while data is being copied to/from some device. DMA requires a special controller, which handles all I/O requests for the CPU.

   (d) Device-independent OS software
   Answer:
   Device-independent OS software is used to create a layer above the device drivers that is independent of any particular device. This allows other parts of the OS to communicate with this layer without having to know anything about the particular devices. It also allows the OS designers to put much of their device handling code in one place rather than in each particular device driver.

   (e) RAID
   Answer:
   RAID stands for Redundant Array of Inexpensive Disks. It is a family of techniques for improving disk performance, reliability or both through redundancy. Because disks are so cheap RAID is widely used in practice.

2. State three advantages of placing functionality in a device controller, rather than in the kernel. State three disadvantages.

   Answer:
   Advantages are higher performance by coding algorithms in hardware, simplification of the kernel by moving out part of its code, and increasing of reliability, since a crash due to a bug would not shut down the operating system. Disadvantages are lack of flexibility in the OS, bugs are harder to fix, and it is more difficult to update algorithms.

3. Why is it important to scale up system bus and device speeds as the CPU speed increases?

   Answer:
   Because the communication between a fast CPU and a slow device or between a fast CPU and a device using a slow bus can be a bottleneck for the whole system. Increasing only the CPU speed will decrease the CPU time, but not the I/O time.

4. How does DMA increase system concurrency? How does it complicate the hardware design?

   Answer:
   DMA increases system concurrency by allowing CPU to perform other tasks while it is transferring data between memory and an I/O device. DMA complicates the hardware design because it shares the memory bus with the CPU, and controllers must directly access physical memory at the same time that the CPU may be accessing physical memory. When DMA is accessing the bus, CPU cannot do it, and visa versa. If caching of disk reads is desirable then this must done in the disk controller rather than in the operating system.

5. For each of the following device access scenarios, say whether DMA would likely provide an appropriate transfer strategy, and briefly explain why: (1) reading user input from the keyboard, (2) sending MPEG video from the file system to the network.

   Answer:
   DMA is appropriate for (2), but not for (1). User input from the keyboard involves a very small amount of data, is asynchronous, and performed at low speed.

   On the other hand, sending MPEG video from the file system to the network, in general, involves transferring a large amount of data, which would require a lot of CPU overhead without DMA. DMA increases system concurrency by allowing the CPU to perform other tasks

6. A floppy disk has 40 cylinders. When storing a file on the floppy disk, the operating system (probably the floppy device driver) often attempts to put the blocks of the file in cylinders that are close to one another (if this can be accomplished). Assume that if the operating system does not attempt to cluster related blocks, two blocks that are logically consecutive are stored about 13 cylinders apart, on average. However, if the operating system makes an attempt to cluster related blocks, the mean inter-block distance can be reduced to 2 cylinders. How long does it take to read 100 block file in each case, if a seek takes 6 milli-seconds per cylinder moved, the rotational latency is 20 milli-seconds on average, and the transfer time is 15 milli-seconds per block?

   Answer:
   Total time = seek time + rotational time + transfer time

   In the first case: 13*6*100 ms + 20*100 ms + 15*100 ms = 11300 milliseconds = 11.3 seconds

   In the second case: 2*6*100 ms + 20*100 ms + 15*100 ms = 4700 milliseconds = 4.7 seconds

7. Are disk scheduling algorithms other than first-come, first-served scheduling useful in a single-user environment? Explain your answer.

   Answer:
   The disk scheduling algorithm depends not on the number of users but the number of processes. In a single-user environment where there is very little multi-processing, the queue of pending requests will usually have length 1 most of the time, so FCFS is likely the most economical scheduling. In a single-user environment with lots of running processes, a more efficient algorithm will be necessary such as the elevator technique.

8. Suppose that a disk drive has 5000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 143, and the previous request was at cylinder 125. The queue of pending requests, in FIFO order, is 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130. Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests, for each of the following disk-scheduling algorithms?

   (a) First-come, first-served.

   Answer:
   Scheduled order: 143, 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130
   Seek times: 57, 1384, 557, 861, 826, 561, 487, 728, 1620
   Total seek time: 7081

   (b) Shortest seek time first.

   Answer:
   Scheduled order: 143, 130, 86, 913, 948, 1022, 1470, 1509, 1750, 1774
   Seek times: 13, 44, 827, 35, 74, 448, 39, 241, 24
   Total seek time: 1745

   (c) Elevator.

   Answer:
   Scheduled order: 143, 913, 948, 1022, 1470, 1509, 1750, 1774, 130, 86
   Seek times: 770, 35, 74, 448, 39, 241, 24, 1644, 44
   Total seek time: 3319

9. Requests are not usually uniformly distributed. For example, some files are accessed more frequently than other files. For instance, files that contain meta-information about the file system structure are accessed much more frequently than a file that contains data. Suppose that you know that 50 percent of the requests are for a small, fixed number of cylinders.

   (a) Would any of the scheduling algorithms discussed in class be particularly good for this case? Explain your answer.

   Answer:
   (There are several possible right answers to this question but here is one.) Shortest seek time first would take more advantage of that situation because requests for the same cylinder would be processed consecutively, and the disk arm would further move to the closest cylinder, minimizing the total seek distance.

   (b) Does your solution in (a) suffer from starvation. If so, can it be modified to prevent starvation?

   Answer:
   It does suffer from starvation. It can be modified to take into account a priority that accounts simultaneously for both age and seek time (using aging). As a request gets older its priority goes up even if it is far away. At some point, its priority becomes higher than requests that have lower seek times and it is selected.

10. Describe how the OS maintains clocks for things such as timesharing, time of day, application alarms, etc. What hardware is required and how is this hardware engineered so that the OS can exploit it?

    Answer:
    The hardware uses a crystal oscillator to generate periodic pulses that can be used to synchronize a processor. The rate of the pulses (perhaps magnified by some small amount) is equal to the cycle time of the machine. In addition, a counter is used to generate an interrupt every n cycles where n is some large number (specified by the OS). This interrupt transfers control to the OS, which can perform its timer management.

    The time between interrupts is usually too short for most timers so the OS creates timers with longer periods but incrementing a counter (in software) every time the interrupt goes off. It can also keep track of multiple timers in software to be used for various things such as time sharing, application alarms, etc. It does this by putting each in a linked list ordered by expiration time.