Chris Kasper
CS 370-001
Homework Set #4

**<u>Chapter 13</u>**

**2. Consider an RS-232 controller with no interrupt capability. If we are to support a 19,200 bps data rate with seven data bits, even parity, and one stop bit, how often should we poll the controller? If each polling operation takes 200 µS, what fraction of the system's time is spent polling?**

Even parity bit = 1 when data bits + parity bit equal even number of 1s

Total bits transmitting = 1 (start bit) + 7 (data bits) + 1 (parity bit) + 1 (end bit) = 10 bits
19,200 bps / 10 bits per byte = 1,920 bytes per second

We should poll the controller every 520.83 µS (1/1920bps)

Fraction of the system's time is polling = 200 µS / 520.83 µS = 38.4%

**3. What is the average access time for a disk drive that spins at 3600 RPM with an average seek time of 50 mS? What if the drive is spun at 10,000 RPM? Does the speed of rotation make a significant difference? What if the drive technology advances and makes the average seek time 9 mS?**

Average rotational latency = ½ of the average time for full rotation

<u>At 3600 RPM:</u>
3600 RPM => 60/3600 for one rotation in seconds = 16.66 mS
This means average rotational latency = 8.33 mS
Average access time = 8.33 mS + 50 mS = 58.33 mS

<u>At 10000 RPM:</u>
10000 RPM => 60/10000 for one rotation in seconds = 6 mS
This means average rotational latency = 3 mS
Average access time = 3 mS + 50 mS = 53 mS

At a 50 mS average seek time, the speed of rotation makes a little bit of difference, it won't make a significant difference in our average access time. As the RPM increases, it will just reduce the rotational latency, and just become a smaller number. The seek time dominates the average access time. At a 9 mS seek time, the rotational latency accounts for more of the average access time, and as the RPM increases, it will diminish the latency, which leads to a decreased average access time. However, like the previous example, the seek time will dominate the average access time as the RPMs increase. So there is a limit.

**6. If an Ethernet controller does not use DMA but generates an interrupt for every byte, and if each interrupt takes 10 µS to process, then can the system effectively use the 10 Mbit/S data rate? Why or why not?**

10 Mbit/S = 1,250,000 bytes/S = 1.25 MB/S
= 0.8 µS per byte

Compared to the 10 µS interrupt processing time, the system cannot effectively use the 10 Mbit/S data rate.

**11. Is the elevator algorithm of any value when used with disk drives that do bad track forwarding and that provide block caches? Why or why not?**

The elevator algorithm won't utilize the block caches, but could be slowed down by bad track forwarding, as the track could be in the wrong spot of the disk then the algorithm was expecting.

**<u>Chapter 15</u>**

**2. In the extended binary coded decimal interchange code (EBCDIC), the letters are not contiguous (for example, there are character values between the letters "i" and "j"). However, the difference between an uppercase letter and the corresponding lowercase one is still a constant. Would the technique for handling the Caps Lock, used in i8042intr ( ), still work for EBCDIC? Why or why not?**

No. This is mainly because the letters are not contiguous. They are grouped in sections of 9 characters, with spaces for other characters in between. This could lead to the wrong character being interpreted.

**4. Using old-style cylinder-head-sector addressing, consider a disk with 1024 cylinders, 4 two-sided platters, and 34 sectors per track. What is the total number of sectors on this disk? What are the cylinder, head, and sector number corresponding to block number 100,000?**

Total number of sectors = 4*2 (platters) * 1024 (cylinders) * 34 (sectors per track) = 278528

CHS address works in the following way: C/H/S …. 10 bits/8 bits/6 bits
Block number = 100,000 = 11000011010100000 in binary

0000000110/00011010/100000
cylinder = 6
head = 26
sector = 32

# Chapter 17

**4. Is it necessary to provide both absolute and relative position seeking service? Why or why not?**

I suppose it is not necessary to provide both an absolute and relative position seeking service. Using a relative position is more of a convenience for the user so they don't have to type the complete path of a file they want to execute that's in a folder or two away from their current working directory.

**5. Suppose we use a list of free blocks where each block is identified by a 32-bit number and each block contains 512 bytes. How many blocks are required to store the free list on a 20-GB disk if the disk is empty?**

Since the block is empty, then every block will contain a pointer to another free block.
So required blocks = 20 GB/ 512 B = 39,062,500

**9. If we have a three-level tree-structured allocation structure as in Figure 17-8, where each block is 1024 bytes and each block pointer is 4 bytes, how large is the largest file that can be represented?**

Block = 1024 bytes -> 1020 bytes for data, 4 bytes for the block pointer
Assuming that the blocks address 256 (1024/4) index blocks each…
Largest file = 1020 * 256 * 256 * 256 = 17,112,760,320 bytes or ~16 GB

**11. Describe how symbolic links could be implemented.**

To implement a symbolic link, we could create a dumby file that points to the absolute path of the directory we want the link to reference. That way when we want to reference a file/function after our directory to be reference, the absolute path of the desired directory can be concatenated with the rest of the path we want.