SAMPLE PROBLEMS

1. Consider a FAT-based (File Allocation Table) file system. Entries in the table are 16 bits wide. A user wants to install a disk with 131072 512-byte sectors.
   a. What is a potential problem?

      Each 16 byte entry in the table is an address of a sector on the disk. The OS can address 65,536 sectors, and the disk has more than that.

   b. Describe a solution to this problem and explain the trade-offs involved.
      Make each FAT entry access a logical sector that is 2 physical sectors. This trades increased internal fragmentation against maintaining the size of the FAT, and backward compatibility.
      Alternatively, you can increase the size of the FAT to be 131,072 17-bit entries, which increases the table size to 278,528 bytes (at least) and increases the complexity of the decoding process. It also ruins any backward compatibility.

2. Generally we've talked about each operating system component in isolation. This question asks you to think about ways in which they interoperate. For each pair of systems below, give a specific way that they interact (or that they could interact). Writing that the file system and I/O system interact because they both use the disk is not worth more than a point, and may be worth none. Writing that the file system and I/O system interact when they determine the mapping from logical blocks $\rightarrow$ physical blocks which impacts the size of file system structures, and the efficiency of the disk usage because larger logical blocks imply more internal fragmentation on the disk is a more complete answer.
   a. How does a demand paged, lazy loaded virtual memory system interact with the process scheduling and creation system?

      A demand paging system will be at its worst at process creation, with an empty address space. The two systems have to work together to pre-fetch a reasonable working set without overloading the paging system by bringing in the entire address space. If the VM system supports shared text (or copy on write), the two need to work together to take advantage of it.

   b. Name another way (not the example above) that the file system and the hard disk drivers in the I/O system interact.

      Feature coordination. If the disk implements interleaving, the file system implementation should not do its own interleaving. The result can be to make both systems worse that useless - actually detrimental. Similar negotiations have to be done for features like disk caching or bad block renaming.

3. On some computer, the clock interrupt handler needs 2 msec (including context switch overhead) per clock tick to execute, and the clock runs at 75 Hz. What fraction of the CPU time is devoted to the clock?

   *The clock interrupt handler is run 75 times every second, which takes 75*2 = 150ms. This is 15% of the CPU time.*

4. List the terms that best describe each of the following:
   a. Operating system code executed when an asynchronous device signals the CPU

      Interrupts service route (ISR)

   b. A type of disk arm scheduling policy

      FIFO, Shortest seek first (SSF), LOOK, C-LOOK, or elevator algorithm

5. In class, we discussed copy-on-write for memory pages shared among multiple processes. We cannot apply this same concept blindly to process creation using Unix fork(), but instead are forced to copy some parts immediately while other parts can be delayed.
   a. Knowing the components of general processes, which parts must be copied immediately, and which parts can be delayed and copied-on-write?

      This question refers to processes, not caching. When we create a new process, we must copy the stack space and registers, but need not copy the entire address space immediately. This is because often a call to fork creates a new address space into which a new process is immediately loaded, making the initial copy a waste of time and space because it is immediately overwritten. An example of this is when a command shell starts up a new process, first doing a fork to create a new copy of itself, but then replacing that copy with the new process which was executed on the command line.

   b. Why is copy-on-write potentially better than copying the entire process immediately upon creation?

      It can save time and space during the process creation (as mentioned above), avoiding the duplicate effort of making a copy, then immediately overwriting it.

6. Briefly describe the steps taken to read a block of data from the disk to the memory using DMA controlled I/O.

   1. *CPU programs the DMA controller by setting its registers so it knows what to transfer where.*
   2. *The DMA controller initiates the transfer by issuing a read request to the disk controller.*
   3. *The disk controller fetches the next word from its internal buffer and writes it to the memory.*
   4. *When the write is finished, the disk controller sends an acknowledgement to the DMA controller*
   5. *If there are more data to transfer (counter is greater than 0), then the DMA controller repeats steps 2 through 4. If all the data has been transferred, the DMA controller interrupts the CPU to let it know that the transfer is complete*

7. Explain what is symbolic link and list at least two of its drawbacks.
   Symbolic link is a way to achieve file sharing. When we want to share a file, we create a new file, or a link. This file contains the path name of the file that we want to share. Later when we access this link file, the operating system sees that the file being accessed is a link, so it can lookup the shared file and all accesses go to that file. Symbolic link is very useful in file sharing. But it also has some drawbacks. First, it needs some disk space for its i-node and data. Second, when

we open a symbolic link, we have to first read the path name in that link, and then follow the link to the shared file. This needs some extra processing time. These two drawbacks are not present in a hard link. Both symbolic link and hard link have another problem, since a file may now have more than one path name, the backup of a file system must be done more carefully.