--- CS 414 Homework 5 ---

1. Disk scheduling

Consider a (small) disk drive with 2500 cylinders, numbered 0 through 2499. The drive is currently serving a request at cylinder number 71, and the previous request was at cylinder 62. The queue of pending requests in FIFO order is:

43, 735, 456, 887, 474, 754

Starting from the current head position, what's the total distance in cylinders that the disk arm moves to satisfy all pending requests for FCFS, SSTF, SCAN, LOOK, C-SCAN and C-LOOK.

2. RAID

Assume that you are writing a data storage application which must maintain k disks of data, but cannot communicate with the global internet or, specifically, offsite data backups. Moreover, you find yourself in the unenviable position of being exposed to intermittent sunspots - specifically, the chance of a neutron striking sectors on your disks and altering the data on them is much, much higher than usual! To improve the reliability of your application, you elect to have it use stable storage to detect and, hopefully, correct disk errors. Cost is not necessarily an object, you can easily afford up to 2^k disks, if necessary.

a) If it must support multiple concurrent readers and multiple concurrent writers, which level(s) of RAID could you use? What are the downsides of each?

b) If it must support multiple concurrent readers but need no longer support multiple concurrent writers, which level(s) of RAID could you now use that you could not use before? What are the downsides of each?

c) Assume (b) again, but now cost has become an issue - you can only afford 2*k-1 disks, and a very crude disk controller. Which levels of RAID become unavailable? Which levels of RAID become undesirable?

3. Free space management

a) Given a disk of size S with blocks of size B, what is the size in bytes of the free-space bitmap for this disk?

b) Assume the free space is represented by a separate linked list, i.e. the pointers are not in the free blocks themselves, but the list is stored in its own space. What is the size in bytes of the list for the above disk, given that the number of free blocks is F? Each linked list entry contains a 32-bit pointer to a free disk block, together with a 32-bit pointer to the next linked list entry.

c) At what point (how full should the disk be, in percent) does it become advantageous to use a linked list instead of a free-space bitmap for this disk?

d) Due to a crash, the free-space linked list is lost, or in an inconsistent state. Can the system reconstruct the free-space list? Explain your answer.

4. Hard and soft links

A file system contains a file called *a.out*. Assume *program.exe* is a hard link and *run.lnk* is a soft link to this file.

a) Suppose we rename *a.out* to hello.exe. What will happen if we try to run *program.exe*? What about *run.lnk*?

b) Suppose we replace *a.out* with a newer version. What will happen if we try to run *program.exe*? What about *run.lnk*?

5. File permissions

Suppose a TA puts a file with midterm scores onto a public network drive, so that students can see their grades, but he forgets to deny write permissions. A moment later, he realizes this, makes the file read-only and is relieved to see that nobody changed the file in the meantime. However, the next day he finds that the file has been changed. How is this possible?

6. Disk caching

a) List some pros and cons of caching disk blocks in main memory.

b) One study of Microsoft Windows NT showed that many files are deleted within milliseconds after being created, and this is far from unusual: perhaps 50% of all files created are quite transient, being deleted again within milliseconds. How might you exploit this knowledge in designing a disk cache management strategy?

c) Database designers sometimes argue that there should be a set of system calls to give programs total control over which disk blocks are held in the file system cache, which ones are written back to disk promptly, and even when to prefetch – in effect, they favor an "external" policy for file system cache management. Thinking about database systems, or other applications that manage complicated on-disk data structures, give some pros and cons of adding such system calls.

7. Fragmentation

a) Why does (external) fragmentation occur in a file system? Why is it undesirable?

b) Suppose you want to design your file system so that fragmentation is automatically kept low and running a defragmenter is not necessary. Describe briefly how you would do this. You can assume there is a reasonable amount of free space (e.g. 20%).

c) Assume a disk is full; is it still possible to defragment it?