Home Work Assignments on I/O and File Management

Q:With DMA (Direct Memory Access):
(a) The processor can read or write directly to a device.
(b) The kernel can read or write directly to a process' memory without intermediate buffers.
(c) A process can read or write to kernel memory without intermediate buffers.
(d) The device can read or write directly to the system's memory.

Q: What is the maximum size disk of the disk that has the follwoing geometry :
Cylinders 1024 ; Heads 16 ; Sectors per track 63

Q: Which of the following strategies is not used in a Disk Scheduling Algorithm?
(a) First in first out (FIFO)
(b) Last in first out (LIFO)
(c) Shortest service time first (SSTF)
(d) Longest service time first (LSTF)
(e) Back and forth over disk (SCAN)

Q: Given a disk with 100 cylinders (0 to 99). Exactly one time unit is required to move the read/write head from one cylinder to the next. At time 0 the heads are cylinder 0 and no requests are pending. The following six requests arrive at tmes shown in:

Time	:	0	10	20	70	80	90
Cylinder	:	21	75	16	68	2	17

Note that once heads are moving arrival of new request will not alter the intended distination of the heads in movement. Assuming, that rotational and transfer times are negligible, give the order in which the cylinders will be visited and the total time required for the Scan algorithm

Q: We have a disc that has 10 tracks per platter with 8 sectors. The drive supports10 writable platters on a common spindle. Each sector stores four 512 byte blocks. There is a read write head for every platter. The heads can be switched in 1ms. The track traversal is sequential and is at the rate of 10ms per track.
a. Show how a 7.5 KB file could be stored ideally.
b. What is the time of retrieval for the file, assuming that (i) the head needs to be switched, (ii) the track needs to be traversed half way to retrieve the first sector of this file.
c. Suppose we decide to fill up this disk with 10 files of 1.1 KB, 20 files of 2.2KB, 30 files of 3.3 KB, and so on. Finally, how many files will be there in the disk? Also, what will be the internal and external fragmentations when the disk is filled up?

Q. In some systems, the i-nodes are kept at the start of the disk. An alternative design is to allocate an i-node when a file is created and put the i-node at the start of the first block of the file. Discuss the pros and cons of this alternative.

Q. Assuming the style of i-node storage where the i-node is stored at the first block of a file, how many disk operations are needed to fetch the i-node for the file /a/b/c/d/e (where a/b/c/d is the absolute directory path, and 'e' is the filename)? Assume that the i-node for the root directory is in the memory, but nothing else along the path is in the memory. Also assume that each directory fits in one disk block.

Q: The FastFile file system uses an inode array to organize the files on disk. Each inode consists of a user id (2 bytes), three time stamps (4 bytes each), protection bits (2 bytes), a reference count (2 byte), a file type (2 bytes) and the size (4 bytes). Additionally, the inode contains 13 direct indexes, 1 index to a 1st-level index table, 1 index to a 2nd-level index table, and 1 index to a 3rd level index table. The file system also stores the first 436 bytes of each file in the inode.

Assume a disk sector is 512 bytes, and assume that any auxilliary index table takes up an entire sector, what is the maximum size for a file in this system. Is there any benefit for including the first 436 bytes of the file in the inode?

Q: Contiguous allocation of files leads to disk fragmentation. Is this internal or external fragmentation?

Q: Consider an indexed file allocation using index nodes (inodes). An inode contains among other things, 7 indexes, one indirect index, one double index, and one triple index.

1. What usually is stored in the inode in addition to the indexes?
2. What is the disadvantage of storing the file name in the inode? Where should the file name be stored?
3. If the disk sector is 512 bytes, what is the maximum file size in this allocation scheme?
4. Suppose we would like to enhance the file system by supporting versioning. That is, when a file is updated, the system creates new version leaving the previous one intact. How would you modify the inode allocation scheme to support versioning? You answer should consider how a new version is created and deleted.
5. In a file system supporting versioning, would you put information about the version number in the inode, or in the directory tree? Justify your answer.

Q:

1. An engineer has designed a FAT-like system and he has used 24 bits for each entry. For a 32-GB disk, what is the minimum size of a file allocation in this system? Justify your answer.

2. Consider an index-based file system with the inode containing 64 indexes, 1 indirect index pointing to a disk block containing an array of direct indexes, and 1 2-level index in the usual way. Assume that each index takes 4 bytes.
   a. What is the maximum file size under this arrangement, if a disk block is 1024 bytes?
   b. Explain how do you compute this maximum size.

3. How many disk accesses does it take to read one disk block at location 3000321 within a file, assuming no caching. Justify your answer.

Q: (a) Consider a very simple file system for a tiny disk. Each sector on the disk holds 2 integers, and all data blocks, indirect blocks, and inodes are 1 disk sector in size (each contains 2 integers). All files stored on disk are interpreted as directories by the file system (there are no "data files"). The file system defines the layout for the following data types on disk:
inode = 1 pointer to a data block + 1 pointer to indirect block

indirect block = 2 pointers to data blocks

directory = a regular file containing zero or more pairs of integers; the first integer of each pair is a file name and the second is the file's inumber

The value "99" signifies a null pointer when referring to a disk block address or directory name. An empty directory has one disk block with the contents "99 99". The inumber for root directory is "/" is 0. The following data are stored on disk:

inode array :

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10 6	7 99		8 99												

Disk blocks :

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	32 3		96 1			1 99	99 99	99 99		57 6					

How many entries can appear in a maximum-sized directory? (Each entry is a pair of integers)

Modify the above data structures to add an empty directory called "87" to directory "/"
