**1.**The open-file table is used to maintain information about files that are currently open. Should the operating system maintain a separate table for each user or just maintain one table that contains references to files that are being accessed by all users at the current time? If the same file is being accessed by two different programs or users, should there be separate entries in the open file table?

By keeping a central open-file table, the operating system can perform the following operation that would be infeasible otherwise. Consider a file that is currently being accessed by one or more processes. If the file is deleted, then it should not be removed from the disk until all processes accessing the file have closed it. This check could be performed only if there is centralized accounting of number of processes accessing the file. On the other hand, if two processes are accessing the file, then separate state needs to be maintained to keep track of the current location of which parts of the file are being accessed by the two processes. This requires the operating system to maintain separate entries for the two processes.

**2.**Some systems provide file sharing by maintaining a single copy of a file; other systems maintain several copies, one for each of the users sharing the file. Discuss the relative merits of each approach.

With a single copy, several concurrent updates to a file may result in user obtaining incorrect information, and the file being left in an incorrect state. With multiple copies, there is storage waste and the various copies may not be consistent with respect to each other.

**3.**What are the advantages of the variation of linked allocation that uses a FAT to chain together the blocks of a file?

Usually, most of the FAT can be cached in memory and therefore the pointers can be determined with just memory accesses instead of having to access the disk blocks.

**4.**Consider a file system on a disk that has both logical and physical block sizes of 512 bytes. Assume that the information about each file is already in memory. For each of the three allocation strategies (contiguous, linked, and indexed), answer these questions:

a. How is the logical-to-physical address mapping accomplished in this system? (For the indexed allocation, assume that a file is always less than 512 blocks long.)

b. If we are currently at logical block 10 (the last block accessed was block 10) and want to access logical block 4, how many physical blocks must be read from the disk?

Let Z be the starting file address (block number).

a. Contiguous. Divide the logical address by 512 with X and Y the resulting quotient and remainder respectively.

1. Add X to Z to obtain the physical block number. Y is the displacement into that block.

2. 1

b. Linked. Divide the logical physical address by 511 with X and Y the resulting quotient and remainder respectively.

1. Chase down the linked list (getting X + 1 blocks). Y + 1 is the displacement into the last physical block.

2. 4

c. Indexed. Divide the logical address by 512 with X and Y the resulting quotient and remainder respectively.

1. Get the index block into memory. Physical block address is contained in the index block at location X. Y is the displacement into the desired physical block.

2. 2

**5.**Consider a file system that uses inodes to represent files. Disk blocks are 8-KB in size and a pointer to a disk block requires 4 bytes. This file system has 12 direct disk blocks, plus single, double, and triple indirect disk blocks. What is the maximum size of a file that can be stored in this file system?

Size of the disk block = 8 KB = 8192 bytes

​Number of bytes required for a pointer to point to a disk block = 4 bytes

Number of pointers per block =8 KB/4 = 8192/4 = 2048

Number of direct disk blocks = 12

As the file has 12 direct disk blocks, plus single, double, and triple indirect disk blocks,

Maximum size = (12×8/KB)+ (2048×8/KB) +(2048×2048×8/KB) + (2048×2048×2048×8/KB) = 68753047648 KB = ​64TB

Thus, the maximum size of a file that can be stored in is file system is 64 terabytes.

**6.**Some DMA controllers support direct virtual memory access, where the targets of I/O operations are specified as virtual addresses and a translation from virtual to physical address is performed during the DMA. How does this design complicate the design of the DMA controller? What are the advantages of providing such a functionality?

Direct virtual memory access allows a device to perform a transfer from two memory-mapped devices without the intervention of the CPU or the use of main memory as a staging ground; the device simply issue memory operations to the memory-mapped addresses of a target device and the ensuing virtual address translation guarantees that the data is transferred to the appropriate device. This functionality, however, comes at the cost of having to support virtual address translation on addresses accessed by a DMA controller and requires the addition of an address translation unit to the DMA controller. The address translation results in both hardware and software costs and might also result in coherence problems between the data structures maintained by the CPU for address translation and corresponding structures used by the DMA controller. These coherence issues would also need to be dealt with and results in further increase in system complexity.

**7.**Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 2150, and the previous request was at cylinder 1805. The queue of pending requests, in FIFO order, is:

2069, 1212, 2296, 2800, 544, 1618, 356, 1523, 4965, 3681

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. FCFS    b. SSTF    c. SCAN

a.13011 b.7865 c.7492

**8.**None of the disk-scheduling disciplines, except FCFS, is truly fair (starvation may occur).

a. Explain why this assertion is true.

b. Describe a way to modify algorithms such as SCAN to ensure fairness.

c. Explain why fairness is an important goal in a time-sharing system.

a. New requests for the track over which the head currently resides can theoretically arrive as quickly as these requests are being serviced.

b. All requests older than some predetermined age could be “forced” to the top of the queue, and an associated bit for each could be set to indicate that no new request could be moved ahead of these requests. For SSTF, the rest of the queue would have to be reorganized with respect to the last of these “old” requests.

c. To prevent unusually long response times.