ID1206: Review Questions 4

Alexander Lundqvist

Fall 2022

1 Question 1

Consider a disk queue with requests for I/O to blocks on cylinders 183, 98, 14, 122, 37, 67, 124, 65, in that order. If the disk head is initially at cylinder 30, what would be disk scheduling using **FCFS**, **SCAN**, and **C-SCAN** scheduling?

1.1 Answer

With **FCFS** strategy the scheduling would be $30 \rightarrow 183 \rightarrow 98 \rightarrow 14 \rightarrow 122 \rightarrow 37 \rightarrow 67 \rightarrow 124 \rightarrow 65$ and the total head movement would be $(183 - 30) + (183 - 98) + (98 - 14) + (122 - 14) + (122 - 37) + (67 - 37) + (124 - 67) + (122 - 65) = 659$.

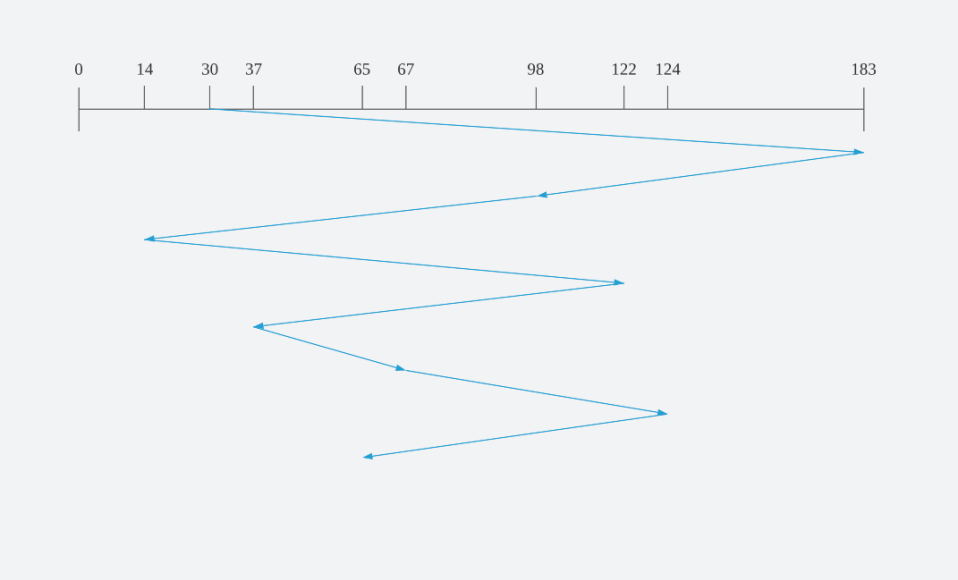

Figure 1: Disk scheduling with FCFS.

With **SCAN** strategy the scheduling would be $30 \rightarrow 14 \rightarrow 37 \rightarrow 65 \rightarrow 67 \rightarrow 98 \rightarrow 122 \rightarrow 124 \rightarrow 183$ and the total head movement would be $(30-14)+(37-14)+(65-37)+(67-65)+(98-67)+(122-98)+(124-122)+(183-124) = 185$.

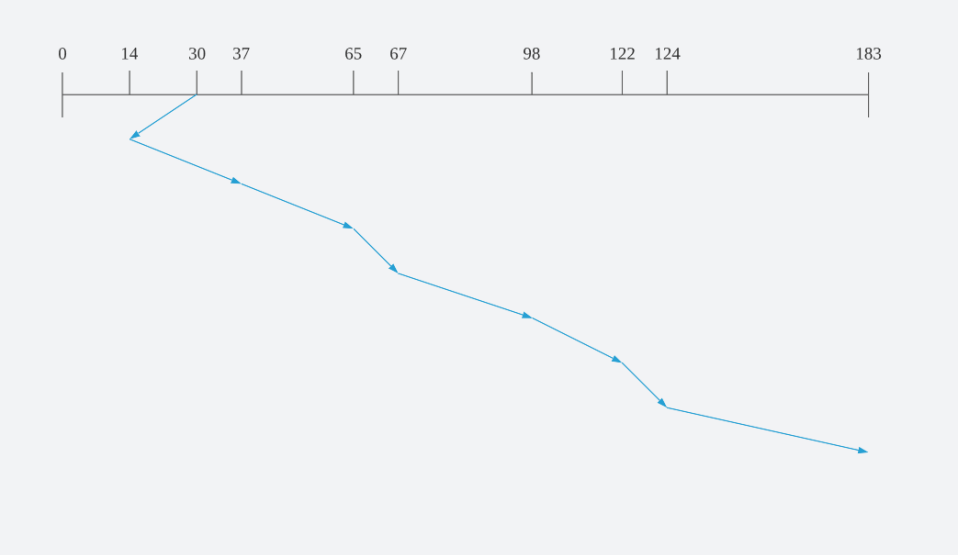

Figure 2: Disk scheduling with SCAN.

With **C-SCAN** strategy the scheduling would be $30 \rightarrow 37 \rightarrow 65 \rightarrow 67 \rightarrow 98 \rightarrow 122 \rightarrow 124 \rightarrow 183 \rightarrow 14$ and the total head movement would be $(37-30)+(65-37)+(67-65)+(98-67)+(122-98)+(124-122)+(183-124)+(183-14)=322$.

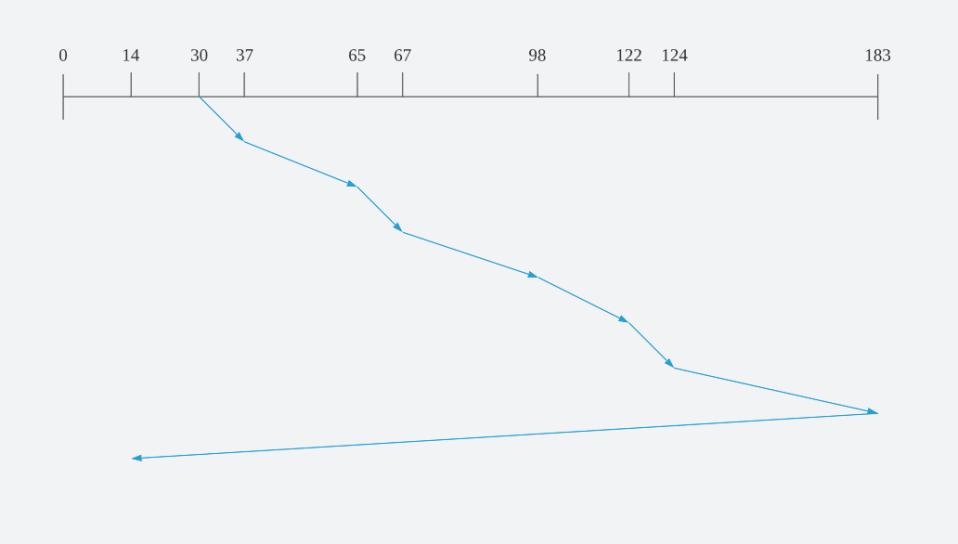

Figure 3: Disk scheduling with C-SCAN.

2 Question 2

Consider a system that supports 5000 users. Suppose that you want to allow 4990 of these users to be able to access one file. How would you specify this protection scheme in UNIX-like (e.g., Linux) OS?

2.1 Answer

The easiest solution to this to create a group with access to this file, then put those 4990 users into that group. However, this solution might not always be feasible. The other way we could accomplish this is by creating an access control list (ACL) with the names of all those users. An ACL is an extension to the standard UNIX permission handling model and is essentially just a list with entries specifying what kind of privileges a group, user or group-specific user has. The entries may concern single files or whole directories.

3 Question 3

Why must the bit map for file allocation be kept on mass storage, rather than in main memory?

3.1 Answer

Since the main memory (or RAM as it is also called) is volatile it will lose its data in the case of loss of power or system crash. This is why we must store the bit map in main memory (A hard drive for instance) to prevent corruption or even total loss of the file allocation data.

4 Question 4

Consider a system that supports the strategies of **contiguous**, **linked**, and **indexed allocation**. What criteria should be used in deciding which strategy is best utilized for a particular file?

4.1 Answer

Contiguous allocation is an easily implemented albeit limited method to allocate files. It should be used for small files because due to how the data is stored, this type of allocation is prone to external fragmentation and thus has a problem when trying to store large files. It can be used for direct access, provided that we use enough information about the location on the disk, but it is more common to use it for sequential access. As such, we can say that for this system, contiguous allocation should be used for small files that are accessed sequentially.

In the **Linked allocation** method, each file is a linked list of memory blocks. As opposed to contiguous allocation, this method is not prone to external fragmentation and is thus suited for both small and large files. However, since the method allows for arbitrary insertion of data and due to the linked list structure (Where you have to iterate over the whole list to find the ith block), it is not feasible with direct access. This method therefore should be used for large (and or small) files which are accessed sequentially.

Indexed allocation solves the issue with direct access in linked allocation by using what is called an index block. Each file gets an index block and it contains pointers to every block that a file is occupying. As a result however, if we would store many small files on the disk we would allocate a large portion of the memory just for these index blocks. The conclusion is that this method should be used for large files that are accessed randomly (and or sequentially).

5 Question 5

Explain how the VFS layer allows an operating system to support multiple types of file systems easily.

5.1 Answer

The virtual file system (VFS) layer works like an interface that abstracts the underlying file systems by providing translation between the operating systems generic file system calls (like open(), read(), write() etc.) and the specific file operations in the individual file systems. In short, applications and kernel level programs do not need to concern themselves with file code for a specific file system.