IOS103

Module 5:
MASS-STORAGE STRUCTURE

Objectives

At the end of the course, the student should be able to:

- *Discuss structure of a disk;*
- *Discuss how the system manage disk spaces;*
- *Define allocation methods used by the system;*
- *Discuss different disk scheduling algorithms such as FCFS, SSTF, SCAN, C-SCAN and LOOK scheduling;*
- *Discuss how the system format a disk, boot from disk and read bad block disk.*

Mass-Storage Structure

Introduction

- The main requirement of secondary storage is to be able to store a very large amount of data permanently.
- One of the earliest secondary-storage media is the magnetic tape. However, it was too slow in comparison with the access time of main memory.
- Magnetic disks provide the bulk of secondary storage for modern computer systems.

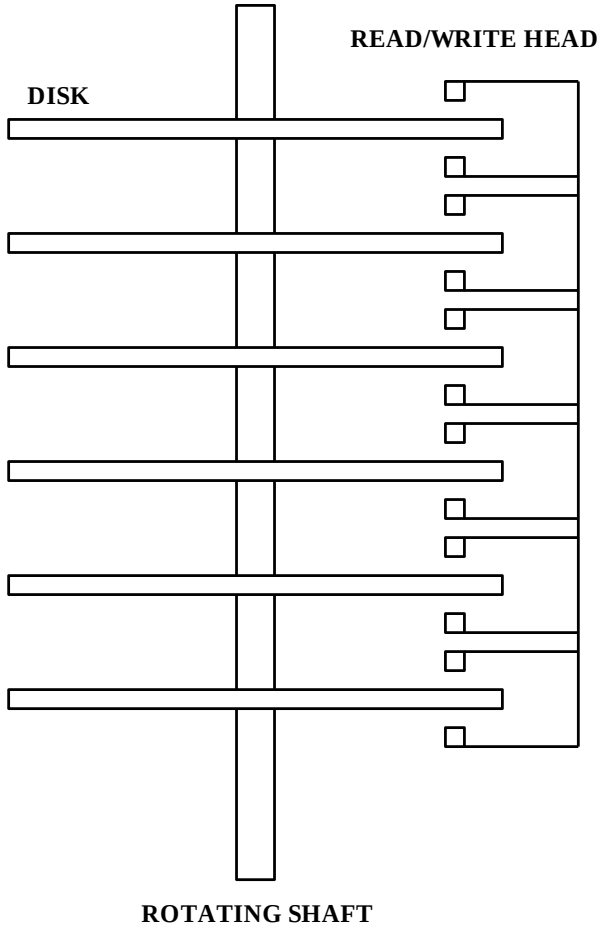
Mass-Storage Structure

Disk Structure

- A magnetic disk system has several disk platters. Each disk platter has a flat circular shape, like a phonograph record. A magnetic material, similar to that of a magnetic tape or floppy diskette, covers its two surfaces. Information is recorded on the surfaces.

Mass-Storage Structure

Disk Structure



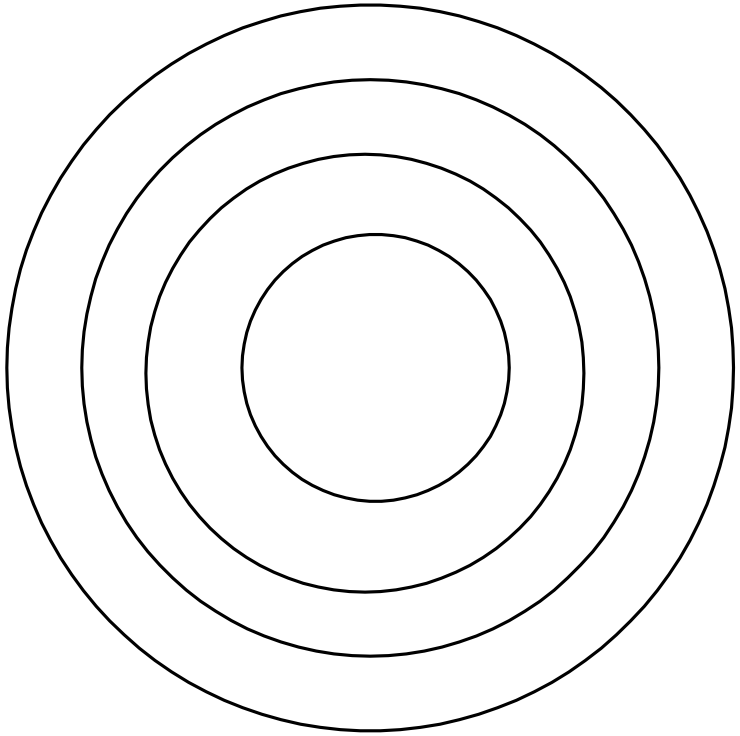
Mass-Storage Structure

Disk Structure

- When the disk is in use, a drive motor spins it at a high speed (for example, 60 revolutions per second). There is a read-write head positioned just above the surface of the platter.
- The disk surface is logically divided into ***tracks***, which are subdivided into ***sectors***. The system stores information by recording it magnetically on the sector under the read-write head.

Mass-Storage Structure

Disk Structure



Mass-Storage Structure

Disk Structure

- There may be hundreds of concentric tracks on a disk surface, containing thousands of sectors. The platter itself may be between 1.8 inches and 14 inches wide. The large sizes are common on large systems because of their higher storage capacities and transfer rates. The smaller sizes are found on PCs, since they have lower cost.
- A ***fixed-head system*** has a separate read-write head for each track. This arrangement allows the computer to switch from track to track quickly, but it requires a large number of heads, making the device extremely expensive.

Mass-Storage Structure

Disk Structure

- A ***movable-head system*** has only one read-write head per surface and the system moves the head to access a particular track (slower but less expensive). In this system, all the tracks on one drive that can be accessed without moving the heads are called a ***cylinder***.

Mass-Storage Structure

Disk Structure

- Disks are rigid metal or glass platters covered with magnetic recording material. Each platter is divided into small sections, and each such section may be changed by the disk head to be in a charged or not charged state. Each section represents a bit. The smaller the changeable sections are, the more bits can be put on a platter and thus the higher the density.
- The head does not actually touch the surface of a disk. Instead, it floats or flies only microns from the disk surface, supported by a cushion of air.

Mass-Storage Structure

Disk Structure

- Head crashes can be problem. If the head contacts the disk surface (due to power failure, for example), the head will scrape the recording medium off the disk, destroying data that had been there.
- Usually, the head touching the surface causes the removed medium to become airborne and to come between the other heads and their platters, causing more crashes. Under normal circumstances, a head crash results in the entire disk failing and needing to be replaced.

Mass-Storage Structure

Disk Structure

- Floppy disks take a different approach. The disks are coated with a hard surface, so the read-write head scans it directly on the disk surface without destroying the data. The coating (and the read-write head) will wear with use, however, and need to be replaced over time.
- Within a track, information is written in blocks or ***sectors***. A sector has a fixed size, specified by the hardware and it is the smallest unit of information that can be read from or written to the disk. Depending on the disk drive, sectors vary from 32 to 4,096 bytes; there are 4 to 32 sectors per track, and from 20 to 1,500 tracks per disk surface.

Mass-Storage Structure

Disk Structure

- The time it takes to access a sector depends on three parameters, the ***seek time***, ***latency time***, and ***transfer time***.

- Seek time is the time it takes to move the read-write head to the correct track while the latency time is the time it takes for the sector to rotate under the head.

- Transfer time is the time it takes to actually transfer data between disk and main memory.

Mass-Storage Structure

Disk Structure

- To access a sector, the system must specify the cylinder or track number, a surface number, and a sector number. Thus, the system views the disk as a three-dimensional array of sectors. However, the OS views this as a one-dimensional array of disk blocks or sectors. Typically, block addresses increase through all sectors on a track, then through all the tracks in a cylinder, and finally from cylinder 0 to the last cylinder on the disk.

Mass-Storage Structure

Disk Structure

- To convert a three-dimensional address (cylinder number *i*, surface number *j*, and sector number *k*) to a one-dimensional block number *b*:

$$b = k + s \times (j + i \times t)$$

where *s* is the number of sectors per track and *t* is the number of tracks per cylinder.

Mass-Storage Structure

Disk Structure

- A disk normally has a device directory indicating which files are on the disk. The directory lists the file by name, and includes such information as where on the disk the file is, and what are its length, type, owner, time of creation, time of last use, protections, and so on.
- The system stores the disk directory on the device, often at some fixed disk address, such as disk address 00001.

Mass-Storage Structure

Disk Structure

- Magnetic disks provide the bulk of secondary storage for modern computer systems. Magnetic tape was used as an early secondary-storage medium, but the access time is much slower than for disks.
- Magnetic tapes are currently used mainly for backup, for storage of infrequently used information, and as a medium for transferring information from one system to another.

Mass-Storage Structure

Disk Structure

- Modern disks are addressed as large one-dimensional arrays of *logical blocks*, where the logical block is the smallest unit of transfer.

- The size of a logical block is usually 512 bytes, although some disks can be low-level formatted to choose a different logical block size, such as 1,024 bytes.

Mass-Storage Structure

Disk Structure

- The one-dimensional array of logical blocks is mapped onto sectors of the disk sequentially. Sector 0 is the first sector of the first track on the outermost cylinder.

- The mapping proceeds in order through the track, then through the rest of the tracks in that cylinder, and then through the rest of the cylinders from the outermost to innermost.

Mass-Storage Structure

Disk Structure

- There are two ways of reading and writing of data on disks:

1. ***Constant Linear Velocity*** (CLV)

In disks using CLV, the density of bits (bits/unit length) per track is uniform. Since the outer tracks are longer, the number of bits on the outer tracks is more than inner tracks. Consequently, there are more sectors on the outer tracks. The drive increases its rotation speed as the head from the outer to the inner tracks to keep the same rate of data moving the head. This method is used in CD-ROM and DVD-ROM drives.

Mass-Storage Structure

Disk Structure

2. ***Constant Angular Velocity*** (CAV)

In disks using CAV, the number of bits per track is uniform (constant number of sectors). This means that the density of bits per track increases from outer tracks to inner tracks. This implies that the disk rotation stays constant. This method is used in hard disks and floppy disks.

Mass-Storage Structure

Disk Structure

- A disk normally has a device directory indicating which files are on the disk. The directory lists the file by name, and includes such information as where on the disk the file is, and what are its length, type, owner, time of creation, time of last use, protections, and so on.
- The system stores the disk directory on the device, often at some fixed disk address, such as disk address 00001.

Mass-Storage Structure

Free Space Management

- Since there is only a limited amount of disk space, it is necessary to reuse the space from deleted files for new files.

- ☞To keep track of the free disk space, the system maintains a ***free-space list***.

Mass-Storage Structure

Free Space Management

Implementation of the free-space list:

1. ***Bit Vector***

A single bit represents each block. If the block is free, the bit is 0; otherwise, the bit is 1.

Example:

- If blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26, and 27 are free, and the rest of the blocks are allocated, then the free-space bit map or vector would be:

Mass-Storage Structure

Free Space Management

11000011000000111001111110001111....

•The main disadvantage of this technique is that the OS often keeps this bit map in main memory.

Mass-Storage Structure

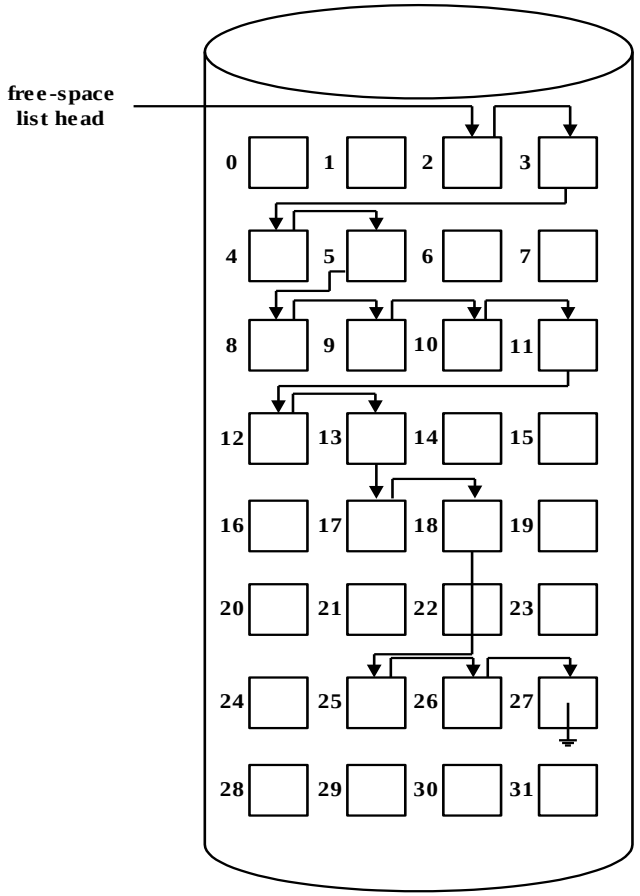
Free Space Management

2. ***Linked List***

•Another approach is to link all the free disk blocks together, keeping a pointer to the first free block. This block contains a pointer to the next free disk block, and so on. This technique however, is not efficient because of the sequential nature of lists.

Mass-Storage Structure

Free Space Management



Mass-Storage Structure

Free Space Management

3. ***Grouping***

- A modification of the free-list approach is to store the addresses of *n* free blocks in the first free block. The first *n*-1 of these are actually free. The last one is the disk address of another block containing the addresses of another *n* free blocks.
- The importance of this implementation is that the addresses of a large number of free blocks can be found quickly.

Mass-Storage Structure

Free Space Management

4. ***Counting***

- Another approach is to take advantage of the fact that, generally, several contiguous blocks may be allocated or freed simultaneously.
- Thus, rather than keeping a list of *n* free disk addresses, the system keeps the address of the first free block and the number *n* of free contiguous blocks that follow the first block. Each entry in the free-space list then consists of a disk address and a count.

Mass-Storage Structure

Allocation Methods

- There are three methods of allocating space to files so that disk space utilization is effective and access time of files is fast.

    1. Contiguous Allocation

    2. Linked Allocation

    3. Indexed Allocation

Mass-Storage Structure

Allocation Methods

1. ***Contiguous Allocation***

- The contiguous allocation method requires each file to occupy a set of contiguous addresses on the disk. If the file is *n* blocks long, and starts at location *b*, then it occupies blocks *b*, *b* + 1, *b* + 2, ..., *b* + *n* - 1. The directory entry for each file indicates the address of the starting block and the length of the area allocated for this file.

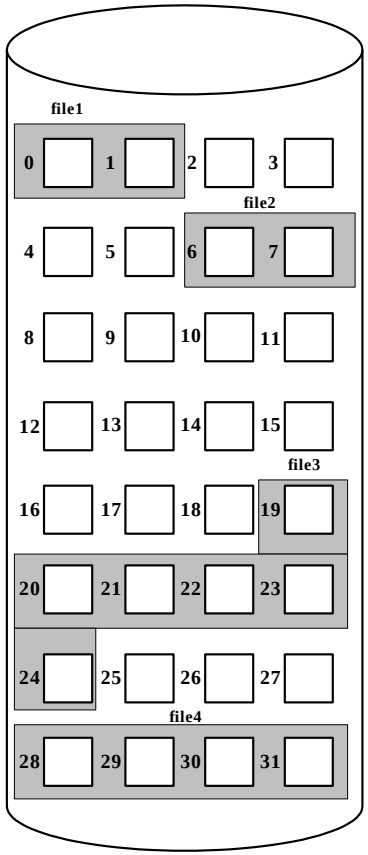
Mass-Storage Structure

Allocation Methods

- The difficulty with contiguous allocation is finding space for a new file. If the file is *n* blocks long, the system must search the free-space list for *n* free contiguous blocks. The actual allocation may follow the first-fit, best-bit, and worst-fit algorithms.
- Each of these algorithms suffers from external fragmentation. External fragmentation exists when enough total free disk space exists to satisfy a request, but this space is not contiguous; storage is fragmented into a large number of small holes. There is also the problem of changes in the size of files.

Mass-Storage Structure

Allocation Methods



directory

filename	start	length
file1	0	2
file2	6	2
file3	19	6
file4	28	4

Mass-Storage Structure

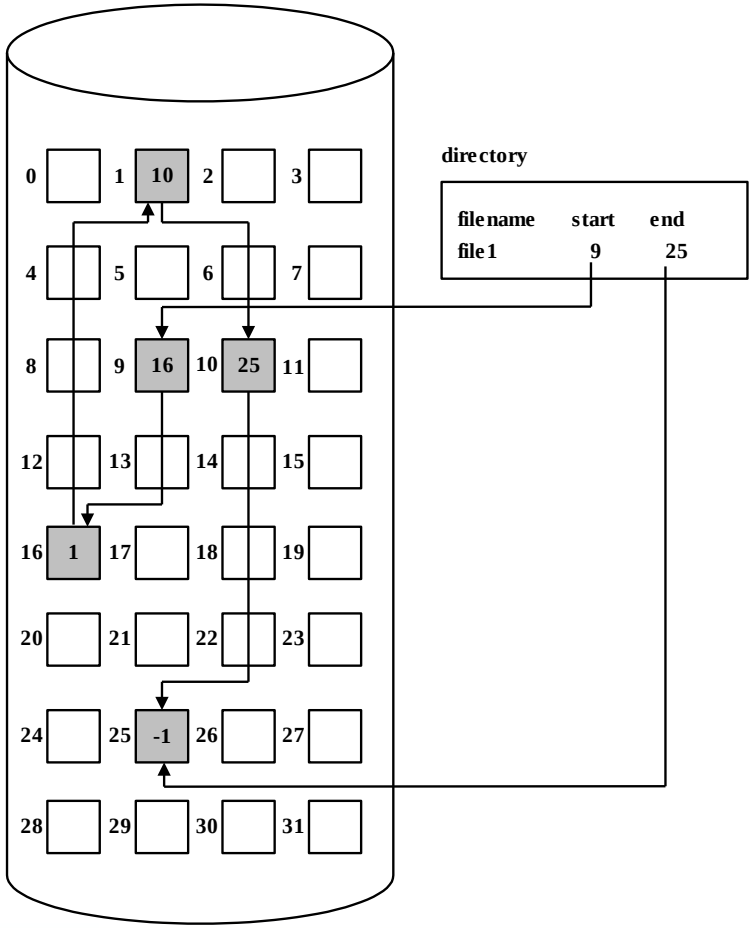
Allocation Methods

2. ***Linked Allocation***

•Each file is a linked list of disk blocks; the disk blocks may be scattered anywhere on the disk. The directory contains a pointer to the first and last blocks of the file. Each block contains a pointer to the next block.

Mass-Storage Structure

Allocation Methods



Mass-Storage Structure

Allocation Methods

- There is no external fragmentation with linked allocation. Any free block on the free-space list can be used to satisfy a request. A file can also continue to grow as long as there are free blocks.
- A disadvantage of this technique is that the pointers also occupy precious disk space. If a pointer requires 4 bytes out of a 512-byte block, then 0.78 % of the disk is being used for pointers, rather than information (for a 40 GB disk, that's 320 MB).

Mass-Storage Structure

Allocation Methods

- The usual solution to this problem is to collect blocks into multiples, called ***clusters***, and to allocate the clusters rather than blocks. For example, the operating system may define a cluster as 4 blocks, and operate on the disk in only cluster units. Pointers then use a smaller percentage on the file's disk space (0.195 %). The cost of this approach is an increase in internal fragmentation, because more space is wasted if a cluster is partially full than when a block is partially full.

Mass-Storage Structure

Allocation Methods

- Another disadvantage is that it is not possible to have direct-access capability. In other words, to access the *i*th block of a file, the system must start at the beginning of the file and follow the pointers until the *i*th block. Each access to a pointer requires a disk read, and sometimes a disk seek.

Mass-Storage Structure

Allocation Methods

- Another problem is reliability. If a pointer were lost or damaged, then it would be difficult, if not impossible, to access the file in its entirety. A solution to this problem is to use doubly linked lists but this increases the overhead for each file.

Mass-Storage Structure

Allocation Methods

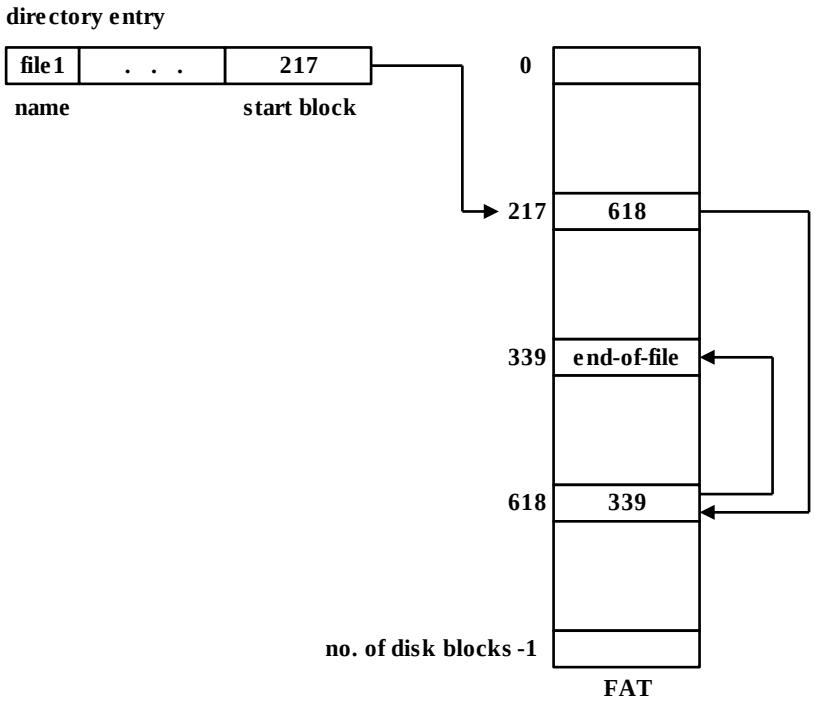
- An important variation on the linked allocation method is the use of the ***file-allocation table*** (FAT). This table has one entry for each disk block, and is indexed by block number. The OS uses this table much like a linked list. The directory contains the block number of the first block of the file. The table entry indexed by that block number then contains the block number of the next block of the file. The chain continues until the last block, which has a special end-of-file value in the table entry. A zero table value indicates free memory blocks.

Mass-Storage Structure

Allocation Methods

Example:

A file consisting of disk blocks 217, 618, and 339.



Mass-Storage Structure

Allocation Methods

3. Indexed Allocation

- Indexed allocation is similar to linked allocation except for the fact that the pointers to the blocks are not scattered but are grouped in one location which is the ***index block***.
- Each file has its own index block, which is an array of disk-block addresses. The *i*th entry in the index block points to the *i*th block of the file. The directory contains the address of the index block.

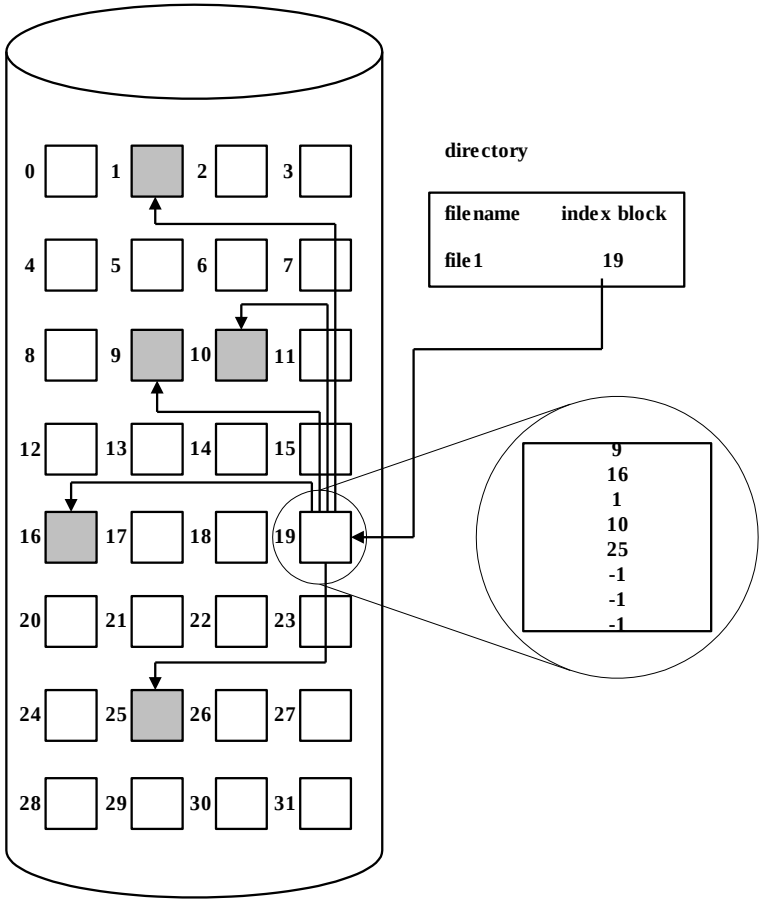
Mass-Storage Structure

Allocation Methods

- Indexed allocation supports direct access, without suffering from external fragmentation. However, it suffers from wasted space. The pointer overhead of the index block is generally greater than the pointer overhead of linked allocation. The OS must allocate an entire index block, even if only one or two pointers will be non-*nil*.

Mass-Storage Structure

Allocation Methods



Mass-Storage Structure

Disk Scheduling

- Since most jobs depend heavily on the disk for loading and input and output files, it is important that disk service be as fast as possible. The OS can improve on the average disk service time by scheduling the requests for disk access.
- If the desired disk drive and controller are available, it can service the request immediately. However, while the drive or controller is serving one request, any additional requests, normally from other processes, should go to a queue.

Mass-Storage Structure

Disk Scheduling

- **Scheduling Algorithms**

1. ***FCFS Scheduling***
2. ***SSTF Scheduling***
3. ***SCAN Scheduling***
4. ***C-SCAN Scheduling***
5. ***LOOK Scheduling***
6. ***C-LOOK Scheduling***

Mass-Storage Structure

Disk Scheduling

1. FCFS Scheduling

- The simplest form of scheduling is the ***first-come first-served*** (FCFS).

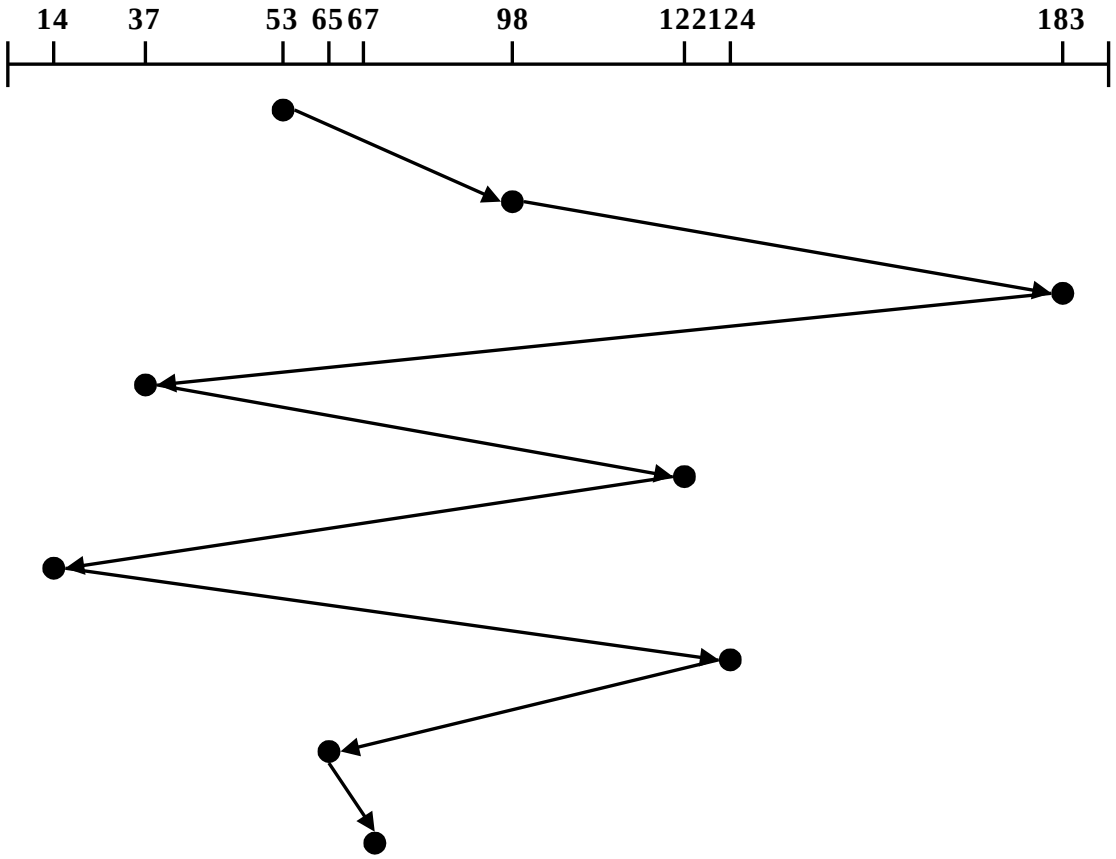
Example:

- Assume that the read-write head is currently at track 53. Consider now an ordered disk queue with requests involving tracks

98, 183, 37, 122, 14, 124, 65, and 67

Mass-Storage Structure

FCFS Disk Scheduling



Mass-Storage Structure

FCFS Disk Scheduling

Computing for the total head movement:

from 53 to 98	=	98 - 53	=	45
from 98 to 183	=	183 - 98	=	85
from 183 to 37	=	183 - 37	=	146
from 37 to 122	=	122 - 37	=	85
from 122 to 14	=	122 - 14	=	108
from 14 to 124	=	124 - 14	=	110
from 124 to 65	=	124 - 65	=	59
from 65 to 67	=	67 - 65	=	2
Total head movement			=	640 tracks

Mass-Storage Structure

FCFS Disk Scheduling

- The problem with this schedule is illustrated by the wild swing from 122 to 14 and back to 124. FCFS may therefore not provide the best (average) service.

Mass-Storage Structure

Disk Scheduling

2. *SSTF Scheduling*

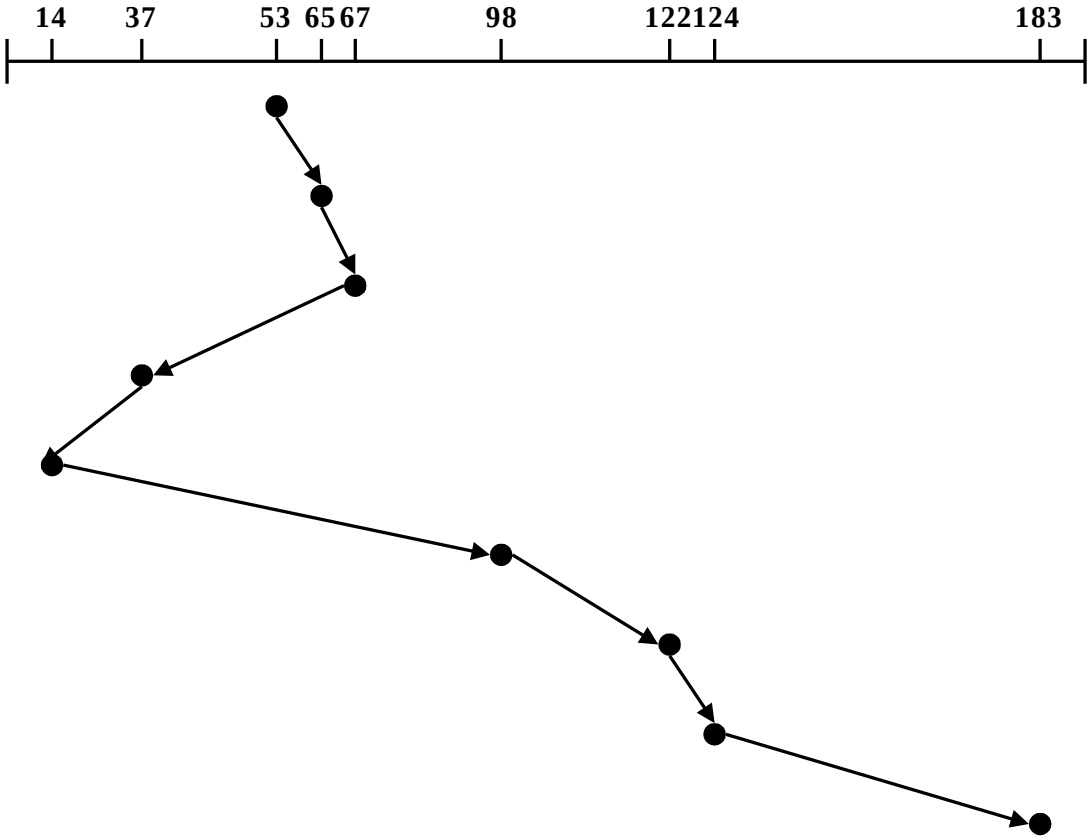
•The ***shortest seek time first*** selects the request with the minimum seek time from the current head position. This means that the system moves the head to the closest track in the request queue.

Example:

Assume that the read-write head is currently at track 53. Consider now an ordered disk queue with requests involving tracks 98, 183, 37, 122, 14, 124, 65, and 67.

Mass-Storage Structure

SSTF Disk Scheduling



Mass-Storage Structure

SSTF Disk Scheduling

Computing for the total head movement:

from 53 to 65	=	65 - 53	=	12
from 65 to 67	=	67 - 65	=	2
from 67 to 37	=	67 - 37	=	30
from 37 to 14	=	37 - 14	=	23
from 14 to 98	=	98 - 14	=	84
from 98 to 122	=	122 - 98	=	24
from 122 to 124	=	124 - 122	=	2
from 124 to 183	=	183 - 124	=	59
Total head movement			=	236 tracks

Mass-Storage Structure

SSTF Disk Scheduling

- This scheduling method results in a total head movement of 236 tracks. This algorithm would result in a substantial improvement in average disk service.
- The problem with SSTF scheduling is that it may cause ***starvation*** of some requests. In theory, a continual stream of requests near one another could arrive, causing the request for a farther track to wait indefinitely.

Mass-Storage Structure

Disk Scheduling

- The SSTF algorithm, although a substantial improvement over the FCFS algorithm, is not optimal.
- For example, if the system moves the head from 53 to 37, even though the latter is not the closest, and then to 14, before turning around to service 65, 67, 98, 122, 124, and 183, then the total head movement would only be 208 tracks.

Mass-Storage Structure

Disk Scheduling

3. *SCAN Scheduling*

•The read-write head starts at one end of the disk, and moves toward the other end, servicing requests as it reaches each track, until it gets to the other end of the disk. At the other end, the direction of head movement reverses and servicing continues. The head continuously scans the disk from end to end.

•The SCAN algorithm is sometimes called the ***elevator algorithm***, since it is similar to the behavior of elevators as they service requests to move from floor to floor in a building.

Mass-Storage Structure

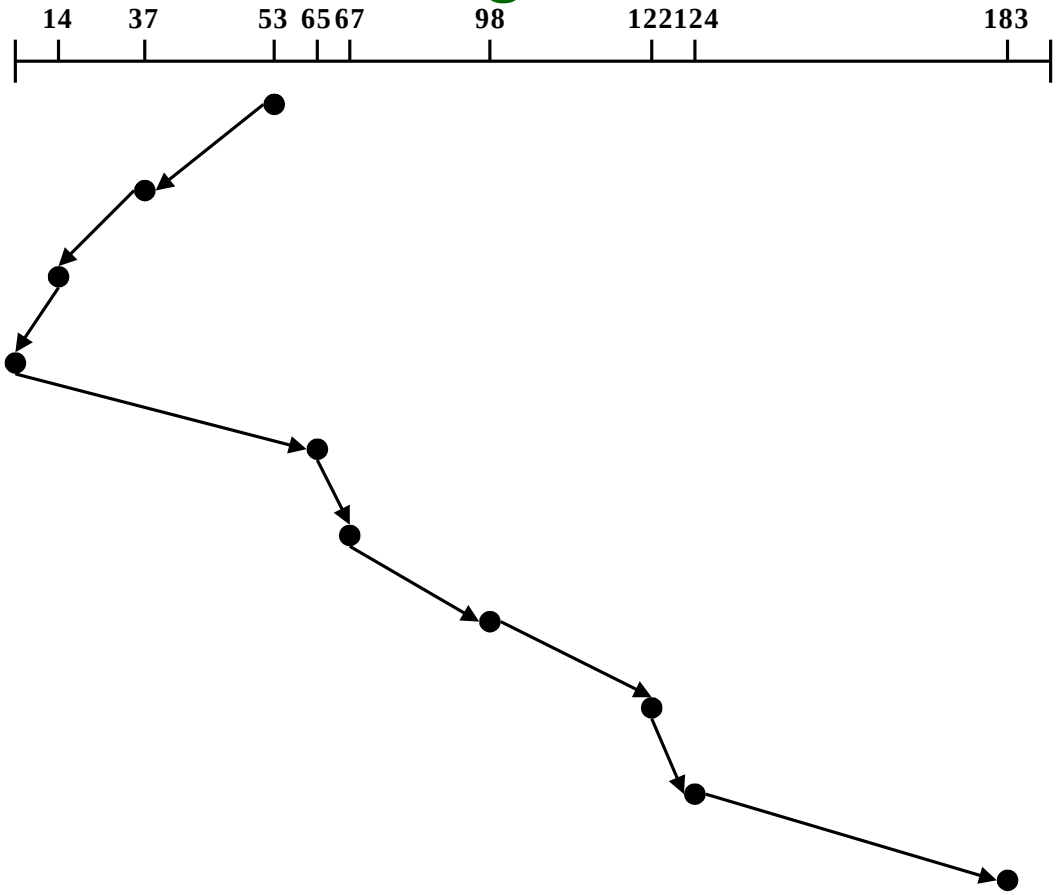
SCAN Disk Scheduling

Example:

Assume that the read-write head is currently at track 53 (going to the direction of track 0). Consider now an ordered disk queue with requests involving tracks 98, 183, 37, 122, 14, 124, 65, and 67.

Mass-Storage Structure

SCAN Disk Scheduling



Mass-Storage Structure

SCAN Disk Scheduling

Computing for the total head movement:

from 53 to 37	=	53 - 37	=	16
from 37 to 14	=	37 - 14	=	23
from 14 to 0	=	14 - 0	=	14
from 0 to 65	=	65 - 0	=	65
from 65 to 67	=	67 - 65	=	2
from 67 to 98	=	98 - 67	=	31
from 98 to 122	=	122 - 98	=	24
from 122 to 124	=	124 - 122	=	2
from 124 to 183	=	183 - 124	=	59
Total head movement			=	236 tracks

Mass-Storage Structure

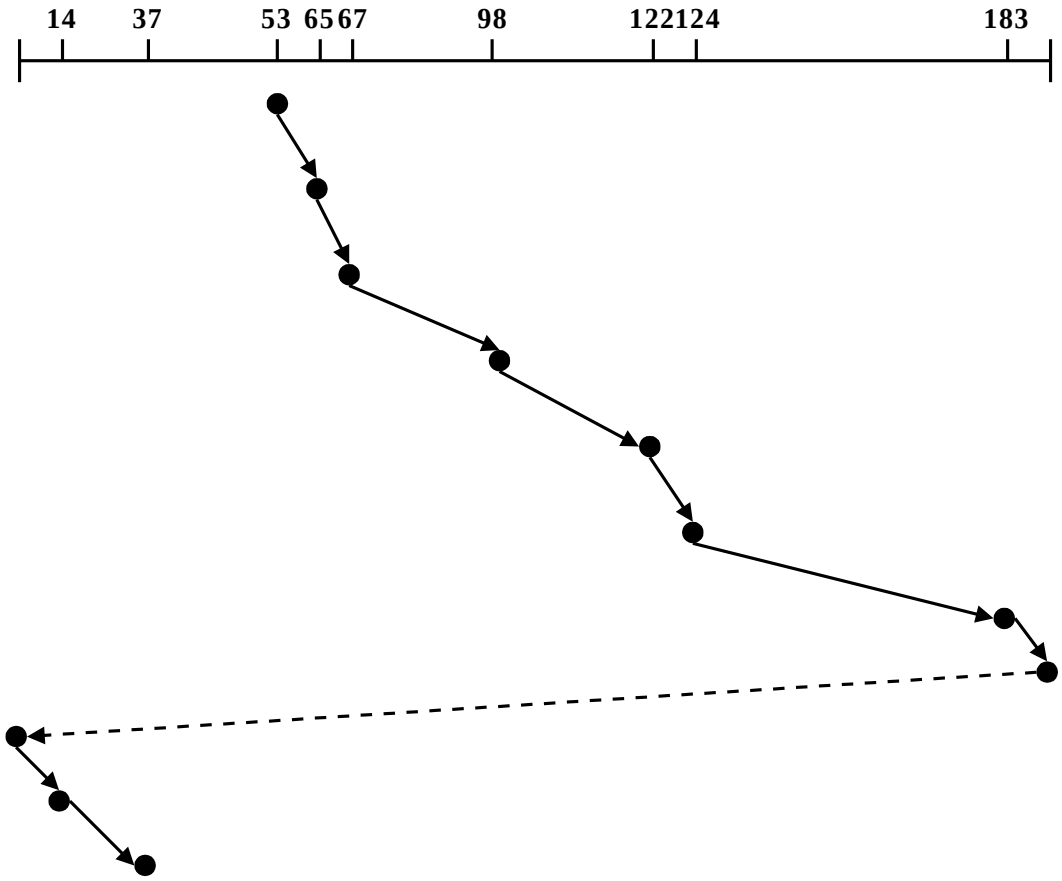
Disk Scheduling

4. *C-SCAN Scheduling*

•The ***Circular-SCAN*** algorithm is a variant of the SCAN algorithm. As does SCAN scheduling, C-SCAN scheduling moves the head from one end of the disk to the other, servicing requests as it goes. When it reaches the other end, however, it immediately returns to the beginning of the disk without servicing any requests on the return trip. C-SCAN scheduling essentially treats the disk as though it were circular, with the last track adjacent to the first one.

Mass-Storage Structure

C-SCAN Disk Scheduling



Mass-Storage Structure

C-SCAN Disk Scheduling

Computing for the total head movement:

from 53 to 65	=	65 - 53	=	12
from 65 to 67	=	67 - 65	=	2
from 67 to 98	=	98 - 67	=	31
from 98 to 122	=	122 - 98	=	24
from 122 to 124	=	124 - 122	=	2
from 124 to 183	=	183 - 124	=	59
from 183 to 200	=	183 - 200	=	17
from 200 to 0	=	200-0	=	200
from 0 to 14	=	14 - 0	=	14
from 14 to 37	=	37 - 14	=	23
Total head movement			=	384 tracks

Mass-Storage Structure

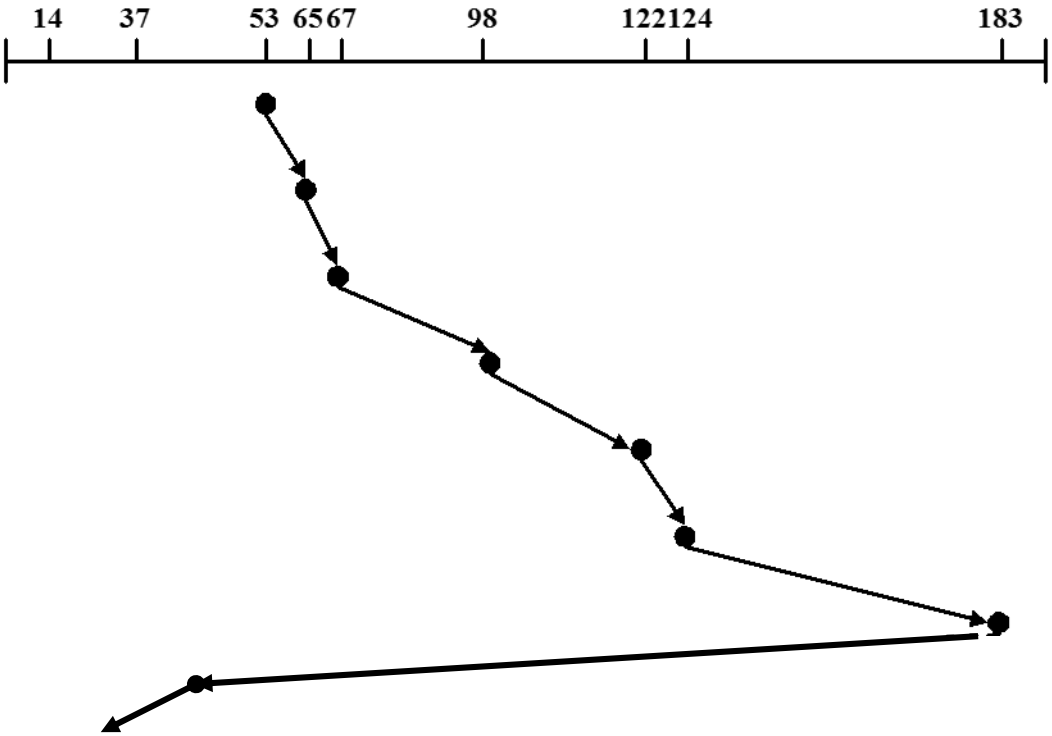
Disk Scheduling

5. *LOOK Scheduling*

•In SCAN and C-SCAN scheduling, the head always moves from one end of the disk to the other. More commonly, the head is only moved as far as the last request in each direction. As soon as there are no requests in the current direction, the head movement is reversed. These versions of SCAN and C-SCAN scheduling are called ***LOOK*** ("look" for a request before moving in that direction) and ***C-LOOK*** scheduling.

Mass-Storage Structure

LOOK Disk Scheduling



Mass-Storage Structure

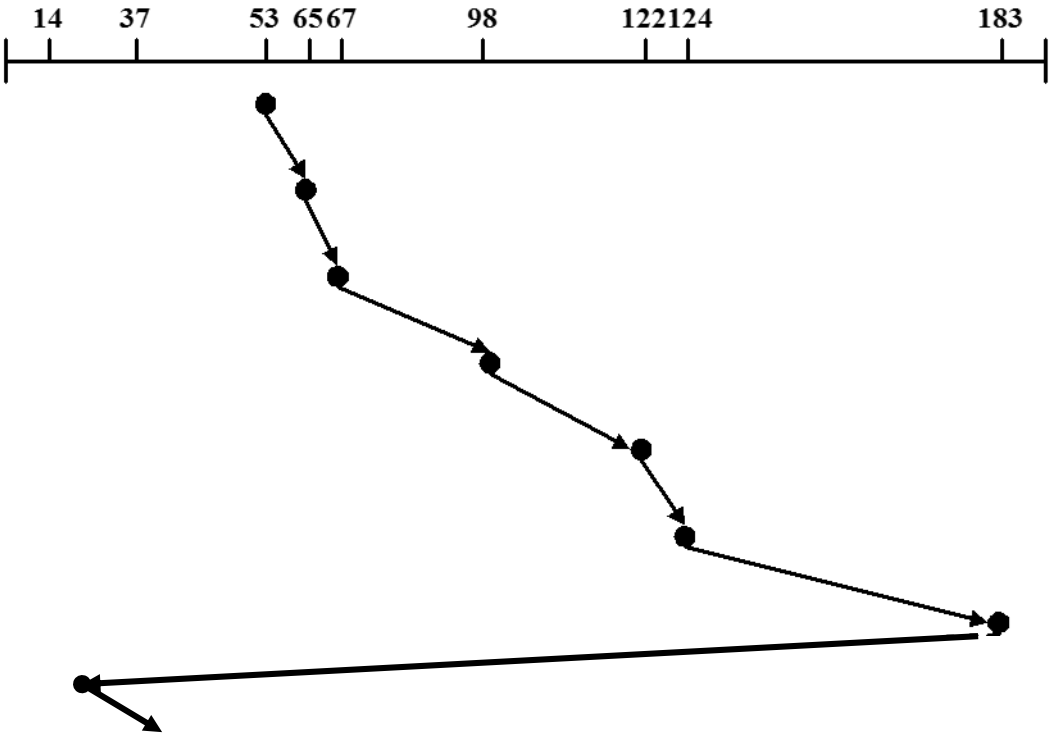
LOOK Disk Scheduling (heading towards right direction)

Computing for the total head movement:

from 53 to 65	=	65 - 53	=	12
from 65 to 67	=	67 - 65	=	2
from 67 to 98	=	98 - 67	=	31
from 98 to 122	=	122 - 98	=	24
from 122 to 124	=	124 - 122	=	2
from 124 to 183	=	183 - 124	=	59
from 183 to 14	=	183 - 37	=	146
from 14 to 37	=	37 - 14	=	23
Total head movement			=	299 tracks

Mass-Storage Structure

C-LOOK Disk Scheduling



Mass-Storage Structure

C-LOOK Disk Scheduling (heading towards right direction)

Computing for the total head movement:

from 53 to 65	=	65 - 53	=	12
from 65 to 67	=	67 - 65	=	2
from 67 to 98	=	98 - 67	=	31
from 98 to 122	=	122 - 98	=	24
from 122 to 124	=	124 - 122	=	2
from 124 to 183	=	183 - 124	=	59
from 183 to 14	=	183 - 14	=	169
from 14 to 37	=	37 - 14	=	23
Total head movement			=	322 tracks

Mass-Storage Structure

Disk Scheduling

Selecting a Disk Scheduling Algorithm

- SCAN and C-SCAN are more appropriate for systems that place a heavy load on the disk.
- If the queue seldom has more than one outstanding request, then all scheduling algorithms are effectively equivalent. In this case, FCFS is a reasonable algorithm (due to its simplicity).

Mass-Storage Structure

Disk Management

There are other aspects of disk management for which an operating system is responsible. They are ***disk formatting***, ***booting from disk***, and ***bad-block recovery***.

Disk Formatting

• Before a computer can make use of a new disk, the disk must be broken into the sectors that the computer can understand. This process is called ***physical formatting***.

Mass-Storage Structure

Disk Management

- When formatted, a disk consists of a set of sectors on each track the head can address. Each sector has a header containing the sector number (so the disk knows when it has found the correct sector) and space for error correcting code (ECC).

Mass-Storage Structure

Disk Management

Boot Block

•For a computer to start running – for instance, when it is powered up or rebooted – it needs to have an initial program to run. This initial program, or ***bootstrap program***, tends to be simple.

•The bootstrap program initializes all aspects of the system, from CPU registers to device controllers to memory contents.

Mass-Storage Structure

Disk Management

- The bootstrap program must also know how to load the operating system and to start it executing. To accomplish this goal, the bootstrap program must locate the operating system kernel, load it into memory, and jump to an initial address.
- Some systems store the bootstrap program in ROM. These systems store a little of the bootstrap in ROM, and store the remainder in the boot blocks at a fixed location on a disk. This disk is known as the ***boot disk*** or ***system disk***.

Mass-Storage Structure

Disk Management

Bad Blocks

•Because disks have moving parts and small tolerances, they are prone to failure. Sometimes, the failure is complete, and the disk needs to be replaced. More frequently, one or more blocks become unreadable and unwritable.

Mass-Storage Structure

Disk Management

- On the IBM PC family with IDE drives, bad blocks must be handled manually. The format command finds bad blocks when the disk is formatted, and writes a special value into the FAT entries to tell the allocation routines not to use those blocks.