UNIT-IV OPERATING SYSTEMS

Unit – IV: File system interface - the concept of a file, access methods, directory structure, file system Mounting, file sharing, protection, file system structure, Mass storage structure - overview of mass storage structure, disk structure, disk Attachment, Disk scheduling.

1Q) Explain the concept of a file?

A) **File:**"A *file* is a space on the disk where logically related information will be stored."*or*"A *file* is a named collection of related information that is recorded on secondary storage device."

File types:A file has a certain defined *structure* according to its type. Different types of file are

a) Text File:A text file is a sequence of characters organized into lines (and possibly pages).

b) Source File:A source file is a sequence of subroutines and functions, each of which is further organized as declarations followed by executable statements.

c) Object File: An object file is a sequence of bytes organized into blocks understandable by the system's linker.

d) Executable File: An executable file is a series of code sections that the loader can bring into memory and execute.

File Attributes:A file is named, for the convenience of its human users and is referred to by its name. A name is usually a string of characters such as "example.c". Some systems differentiate between lower and upper case characters.

A file has certain attributes, which are vary from one operating system to another. They are

Name – The symbolic file name is the only information kept in human-readable form

Type – The information is needed for systems that support different types

Location – This information is a pointer to file location on device

Size – The current file size (in bytes, words or blocks)

Protection – Access-control information controls who can do reading, writing, executing

Time, date, and user identification – The can be useful data for protection, security, and usage monitoring

Information about files are kept in the directory structure, which is maintained on the disk

File Operations: A file is an *abstract data type*. The operating system provides system calls to create, write, read, reposition, delete and truncate files.

Create – Two steps are required to create a file. First, space for the file system must be found for the file. Second, an entry for the new file must be made in the directory.

Write – To write the data into a file, a system call is required for both name and information to be written into the file. The system must keep a *write a pointer* to the location in the file where the next write is to take place. The write pointer must be updated whenever a write occurs.

Read – To read from a file, we use a system call that specifies both name of the file and where the next block of the file should be put. The system keeps a *read pointer* to the location in the file where the next read is to take place.

file seek – Repositioning within file, is also known as a *file seek.*

Delete – To delete a file, we search the directory for the named file.

Truncate – When the user wants the attributes of a file to remain the same, but wants to erase the contents of a file. This function can do this job.

Open(Fi) – search the directory structure on disk for entry *Fi*, and move the content of entry to memory

Close (Fi) – move the content of entry *Fi* in memory to directory structure on disk

To open files, Several pieces of data are needed to manage open files:

File pointer: pointer to last read/write location, per process that has the file open

File-open count: counter of number of times a file is open – to allow removal of data from open-file table when last processes closes it.

Disk location of the file: cache of data access information

Access rights: per-process access mode information

Common File Types: The following table explains common file types:

file type	usual extension	function
executable	exe, com, bin or none	read to run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll, mpeg, mov, rm	libraries of routines for programmers
print or view	arc, zip, tar	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm	binary file containing audio or A/V information

10 2Q) **Write about file access methods?**

A) File Access Methods: File access mechanism refers to the manner in which the records of a file may be accessed. There are several ways to access files –

a. Sequential access

b. Direct/Random access

c. Indexed sequential access

A. **Sequential access:** A sequential access is that in which the records are accessed in some sequence, i.e., the information in the file is processed in order, one record after the other. This access method is the most primitive one. Example: Compilers usually access files in this fashion.

b. Direct/Random access: • Random access file organization provides, accessing the records directly.

• Each record has its own address on the file with by the help of which it can be directly accessed for reading or writing.

• The records need not be in any sequence within the file and they need not be in adjacent locations on the storage medium.

c. Indexed sequential access: • This mechanism is built up on base of sequential access.

• An index is created for each file which contains pointers to various blocks.

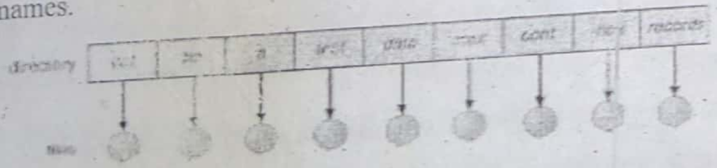
• Index is searched sequentially and its pointer is used to access the file directly.

3Q) Write about Directory Structure?

A) Directory Structure:

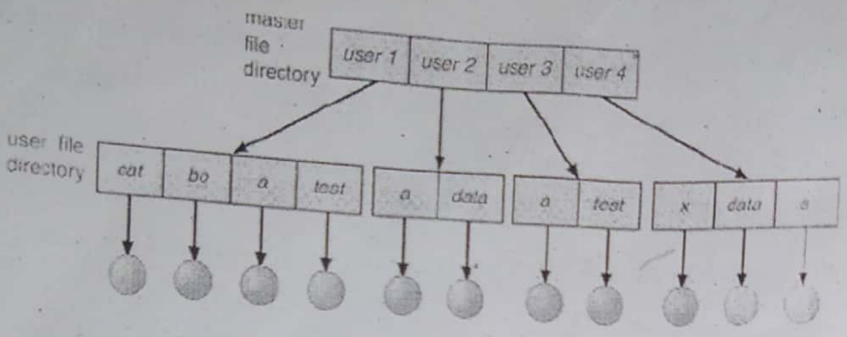
Single-Level Directory: In a single-level directory system, all the files are placed in one directory. This is very common on single-user OS's. A single-level directory has significant limitations, however, when the number of files increases or when there is more than one user.

Since all files are in the same directory, they must have unique names. If there are two users who call their data file "test", then the unique name rule is violated. Although file names are generally selected to reflect the content of the file, they are often quite limited in length. Even with a single-user, as the number of files increases, it becomes difficult to remember the names of all the files in order to create only files with unique names.

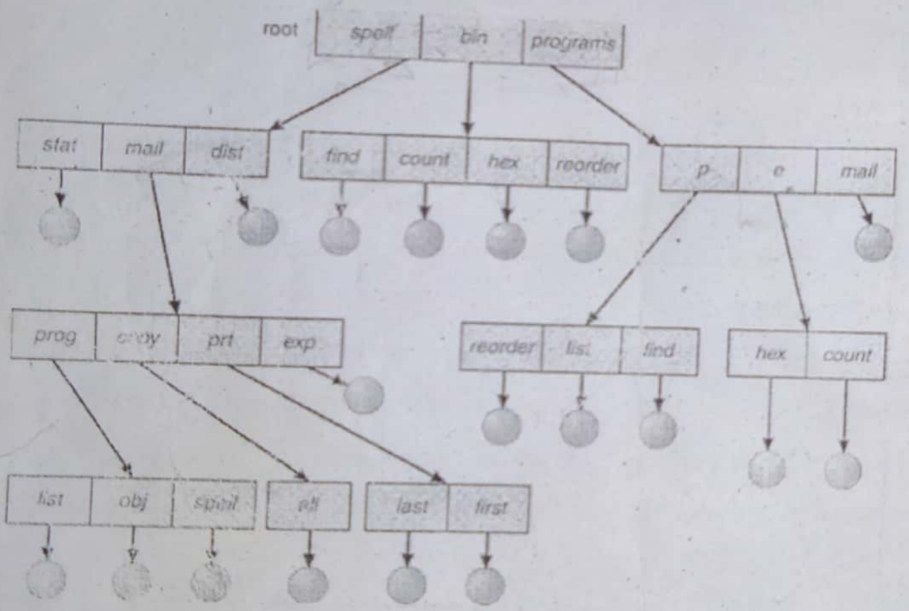


Two-Level Directory: In the two-level directory system, the system maintains a master block that has one entry for each user. This master block contains the addresses of the directory of the users. There are still problems with two-level directory structure. This structure effectively isolates one user from another. This is an advantage when the users are completely independent, but a disadvantage when the users want to cooperate on some task and access files of other users.

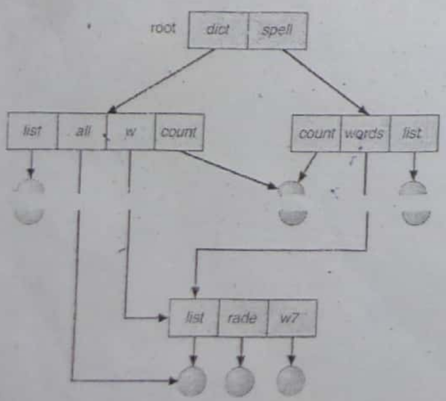
Some systems simply do not allow local files to be accessed by other users.

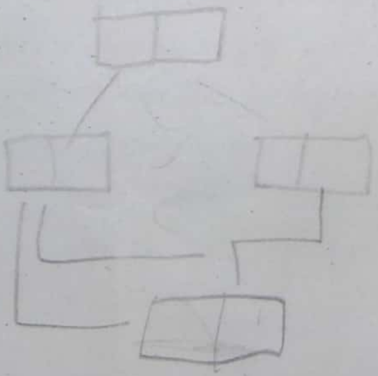


Tree-Structured Directories: In the tree-structured directory, the directory themselves are files. This leads to the possibility of having sub-directories that can contain files and sub-subdirectories. When a request is made to delete a directory, all of that directory's files and subdirectories are also to be deleted.



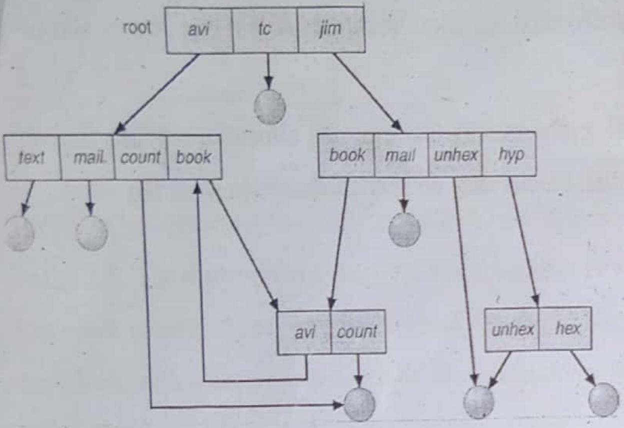
Acyclic-Graph Directories: A tree structure prohibits the sharing of files or directories. An *acyclic graph*, which is a graph with no cycles, allows directories to have shared subdirectories and files. This is explained in the following diagram:





above diagram *count* is shared. The primary advantage of acyclic graph is the relative simplicity of the ithms to traverse the graph and determine when there are no more references to file.

eral Graph Directory: This allows only links to file not subdirectories. Garbage-collection is necessary y because of possible cycles in the graph. The difficulty is to avoid cycles as new links are ded to the structure. This is explained in the following diagram:



4Q) Explain about disk space Allocation Methods in file system implementation?

A) **Allocation Methods:** An allocation method refers to how disk blocks are allocated for files. Three major methods of allocating disk space are:

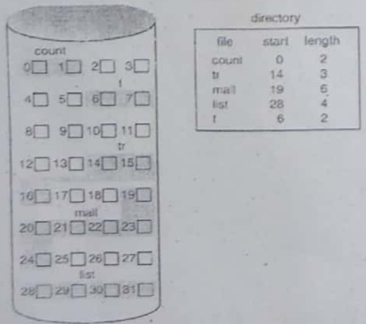
a) Contiguous allocation

b) Linked allocation

c) Indexed allocation

a) Contiguous allocation:

The contiguous allocation method requires each file to occupy a set of contiguous blocks on the disk. This method is simple – only starting location (block # (no.)) and length (number of blocks) are required. Both sequential and random access can be supported by contiguous allocation. One difficulty with contiguous allocation is finding space for a new file. The contiguous disk-space-allocation problem can be seen to be particular application of the general dynamic storage-allocation problem (Wasteful of space). Stored files cannot grow in future. The contiguous allocation of disk space is explained in the following diagram.



file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

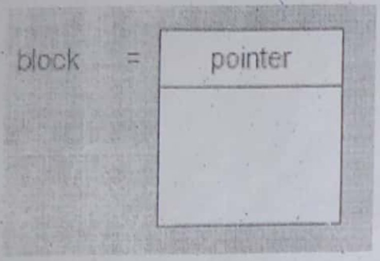
This algorithm suffers from the external fragmentation. As files are allocated and deleted, the free space is broken into little pieces.

A file that grows slowly over a long period (months or years) must be allocated enough space fo
size, even though much of that space may be unused for a long time. The file, therefore, has a large a
of internal fragmentation.

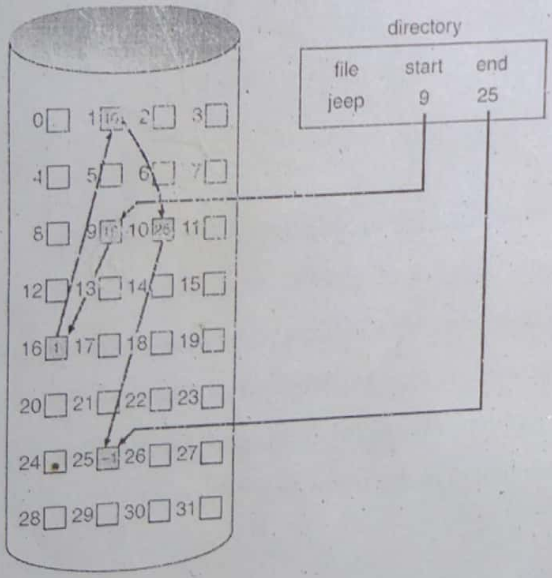
To avoid several drawbacks of contiguous allocation method, many newer file systems use a modifi
contiguous allocation scheme. Extent-based file systems allocate disk blocks in **extents.**

An **extent** is a contiguous block of disks. Extents are allocated for file allocation. A file consists of one or more extents.

b) Linked allocation: Linked allocation solves all problems of contiguous allocation. With linked allocation, each file is a linked list of disk blocks. The disk blocks may be scattered anywhere on the disk.



This scheme is simple. It needs only starting address. For example, a file of five blocks might start at block 9, continue at block 16, then block 1, block 10 and finally block 25. This is explained in the following diagram:

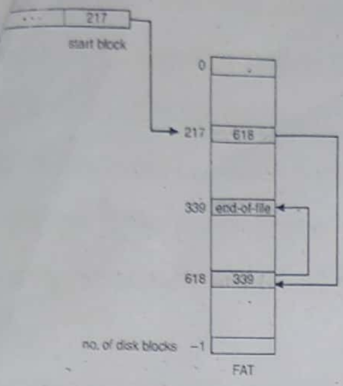


Free-space management system – no waste of space

No random access

Mapping

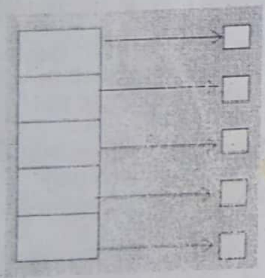
9845225566



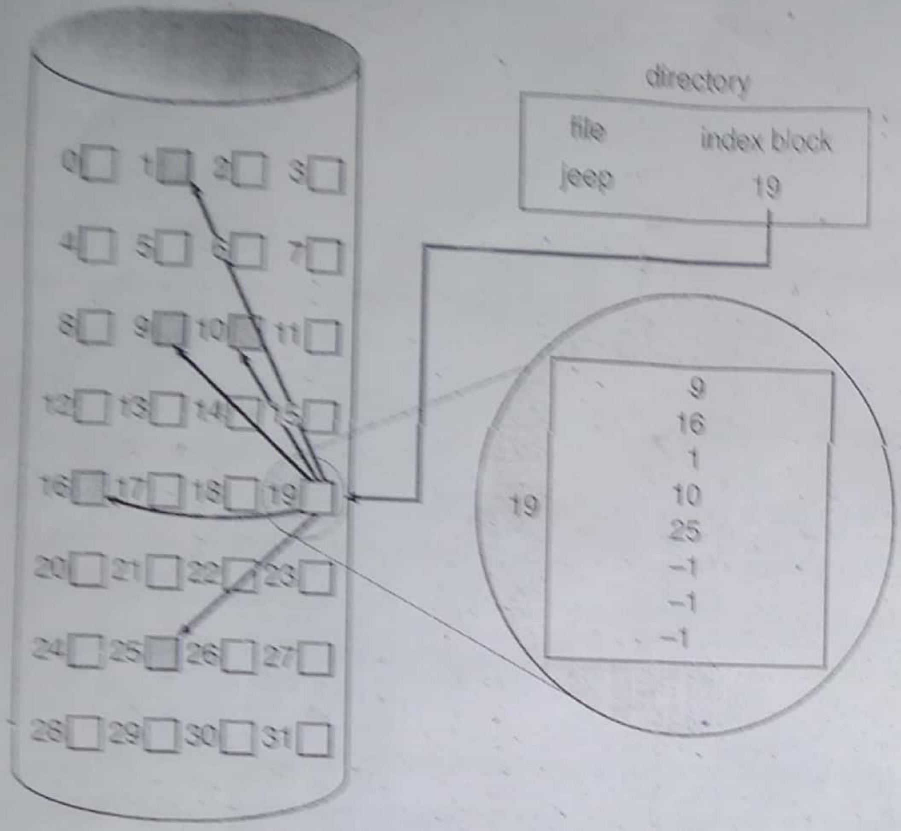
The table has one entry for each disk block and is indexed by block number. The FAT is used much as is a linked list. The directory contains the block number of the first block of the file. The table entry indexed by that block number then contains the block number of the next block in the file. This chain continues until the last block, which has a special end-of-file value as the table entry. Unused blocks are indicated by a 0 (zero) table value.

c) Indexed Allocation: Linked allocation solves the external fragmentation problem and size declaration problems of contiguous allocation. But in the absence of FAT, linked allocation can not support efficient direct access because the pointers to the blocks are scattered with the blocks themselves all over the disk and need to be retrieved in order. Indexed allocation solves the problem in linked allocation by bringing all pointers together into one location is known as ***index block***.

The following diagram shows the logical view of the index table:



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 entry contains the address of the index block. This is explained in the following diagram:



Indexed allocation supports direct access without suffering from external fragmentation. This method suffers from wasted space.

5Q) Write about protection and security?

A) Protection is one of the major role in the OS and file system interface. The protection can be provided in many ways. For example, these are represented as follows.

Types of access: it provides a LIMITEED FILE ACCESS TO THE USER. Several types of operations may be controlled by file system interface is as follows

- Read: read from the file
- Write: write or rewrite the file
- Execute: load the file into memory and execute
- Append: write the new information at the end of the file
- Delete: delete the file and releases the memory
- List: list the name and attributes of the file

Access list and groups:

In the file system interface,every file may requires 3 persons. They are 1.owner 2.group 3.others

Owner – the user who created the file is the owner

Group – a set of users who are sharing the file with in the work group

Universe/others – all other users in the file system interface

In the MS-Dos operating system the access rightes are applied though "attrid command is as follows

:\ Attrib +751<fole name>

the UNIX operating system the access rightes are applied though "chmod" command is as follows

Chmod +751<file name>

The attributes read contains 4,write contains 2,and execute contains 1

Hence in the above example the owner contains read and execute permission, group contains read and execute permissions and others contain execute permission only

Security:

Security requires not only an adequate protection system, but also consideration of the external environment within which the system operates. The system protect it from

i) unauthorized access. ii) Malicious (harmful) modification or destruction

iii) accidental introduction of inconsistency.

It is easier to protect against accidental than malicious misuse.

Authentication: A major security problem for operating systems is ***authentication***. Generally, authentication is based on one or more of three items: user possession (a key or card), user knowledge (a user_id and password) or user attributes (fingerprint, retina pattern or signature). " *Authentication* means Verifying the identity of another entity

– Computer authenticating to another computer

– Person authenticating to a local computer

– Person authenticating to a remote computer

Passwords: The most common authentication of a user's identity is via a user ***password***. In password authentication, users are authenticated by the password.

- User has a secret password.

- System checks it to authenticate the user.

- Vulnerable to eavesdropping when password is communicated from user to system. The big problem with password-based authentication is eavesdropping.

Vulnerabilities : 1) *External Disclosure* – Password becomes known to an unauthorized person by a means outside normal network or system operation. Includes storing passwords in unprotected files or posting it in an unsecured place.

2) *Password Guessing* – Results from very short passwords, not changing passwords often, allowing passwords which are common words or proper names, and using default passwords.

3) *Live Eavesdropping* – Results from allowing passwords to transmit in an unprotected manner.

4)*Verifier Compromise* – Occurs when the verifier's password file is compromised via an internal attack.

5) *Replay* – Recording and later replaying of a legitimate authentication exchange.

There are various ***Techniques*** to handle the passwords safely. They are

1) One-Time Passwords 2) Challenge-Response 3) On-Line Cryptographic Servers

4) Off-Line Cryptographic Servers 5) Use of Non-Repeating Values6) Mutual Authentication Protocols

6Q) Write about free space management?

A)In the file system, it is necessary to reuse the space from deleted files for new files. In the file system the disk maintains the free space list.The free space list is implemented in 4 ways. They are

1. Bit Vector

2.Linked List

3. Grouping

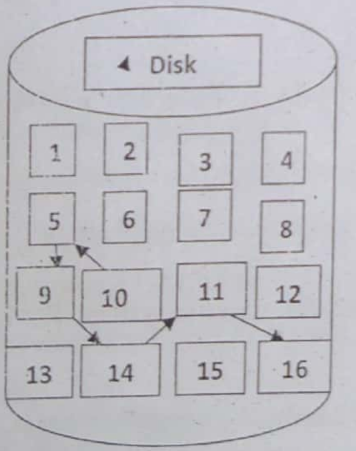
4. Counting

Bit vector: It is also called as Bit map.In this representation ,each block is represented by either '1' or 'o'. If the block is free then it is representd by '1'.other wise Zero

For example conside a disk where the blocks are 2,3,4,8,9,13 are free.then the bit vector is represented as follows.00111000110001

Ihe main advantage of this representation is relatively simple and efficient.In this,the calculation of the block no.is (no of Bits per word)×(no of zero values words)+off set of first one bit.

Linked List: In this representation ,to link together all the free disk blocks with the help of a pointer the linked list is implemented.In the linked list,the first block contains a pointer to the next n disk block and so on.For example,It can be representes as follows



Grouping: In this representation ,every block contains n free blocks.the last block contains the address of another n free blocks.the importance of this implementation is used in the large no of free blocks

Counting: In this representation ,several contiguous blocks may be allocated or free simultaneously .In this representation a disk block maintains the list of n free contiguous blocks.Each entry requires more space than a simple disk address and the count is generally greater than 1.

7Q) Write about file system structure?

A) The functionality of the file is used to store large amount of information. In the file system implementation, the layered approach is represented as follows.

pplication programs
|
Logical file system
|
File organisation module
|
Basic file system
|
I/O control
|
Devices

- The lowest level contains the I/O control consists of device drivers and interrupt handlers to transfer the information between the memory and disk system.
- The file organisation module defines the information about the files, logical blocks as well as the physical blocks. It also include free space manager.
- The logical file system uses the directory structure to provide the file organisation module. It is also responsible for protection and security.
- Application programs are written by the user with the help of languages like c ,c++,java,.., e.t.c. in the application programs, they are various types of files. For example, they are text files, binary files, indexed files, random files, executable files, e.t.c..

ro on

E **8Q) Explain disk scheduling with its algorithms?**

A)1) The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth.

2) Access time has two major components a) *Seek time* is the time for the disk to move the heads to the cylinder containing the desired sector.

b) *Rotational latency* is the additional time waiting for the disk to rotate the desired sector to the disk head.

3) Minimize seek time

4) Seek time □□seek distance

5) Disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.

6) Several algorithms exist to schedule the servicing of disk I/O requests.

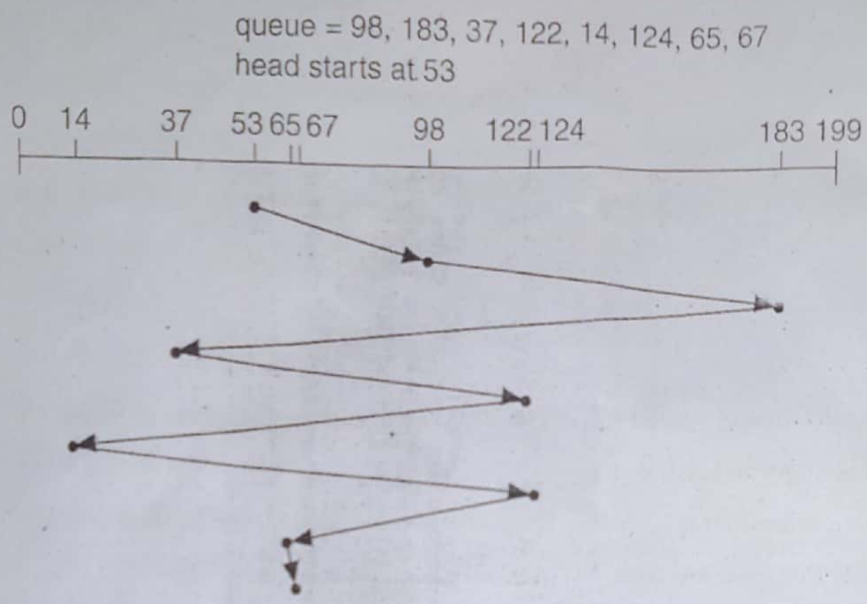
7) We illustrate them with a request queue (0-199).

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

Head pointer 53

1. FCFS Scheduling:

The simplest form of disk scheduling is First-come, First-served (FCFS). This algorithm is basically fair, but it generally does not provide the fastest service.

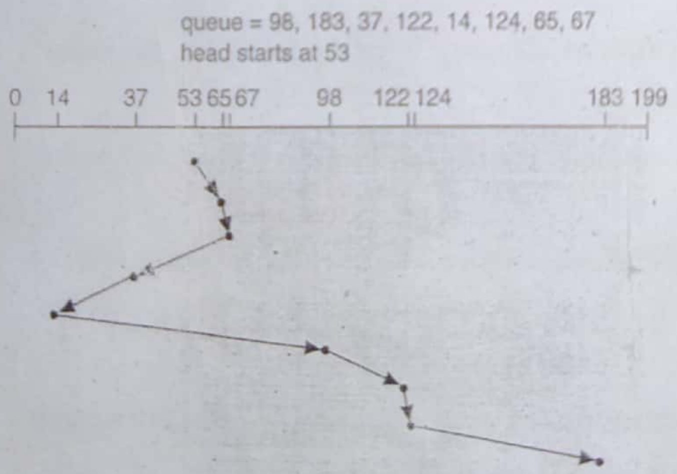


To the above example, total head movement of 640 cylinders.

2. SSTF (Shortest Seek Time First) Scheduling:

a) SSTF selects the request with the minimum seek time from the current head position.

b) SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests.Ex: This shows total head movement of 236 cylinders



3) SCAN Scheduling:

a) In the SCAN algorithm, the disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.

b) Sometimes this algorithm is also called the *elevator algorithm* because the disk arm behaves just like elevator in a building, first servicing all the requests going up and then reversing to service requests the other way.

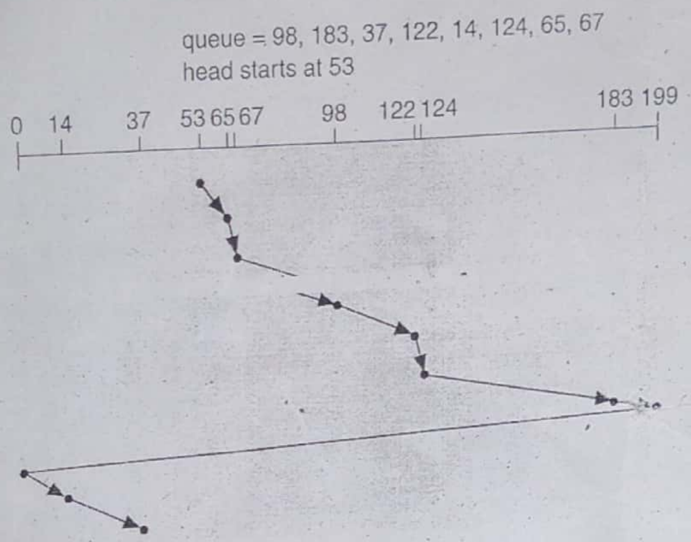
Ex: This algorithm shows total head movement of 208 cylinders.

N Scheduling:

scheduling algorithm provides a more uniform wait time than SCAN.

head moves from one end of the disk to the other, servicing requests along the way. When it reaches other end, however, it immediately returns to the beginning of the disk, without servicing any requests on return trip.

) The C-SCAN scheduling algorithm treats the cylinders as a circular list that wraps around from the last cylinder to the first one.

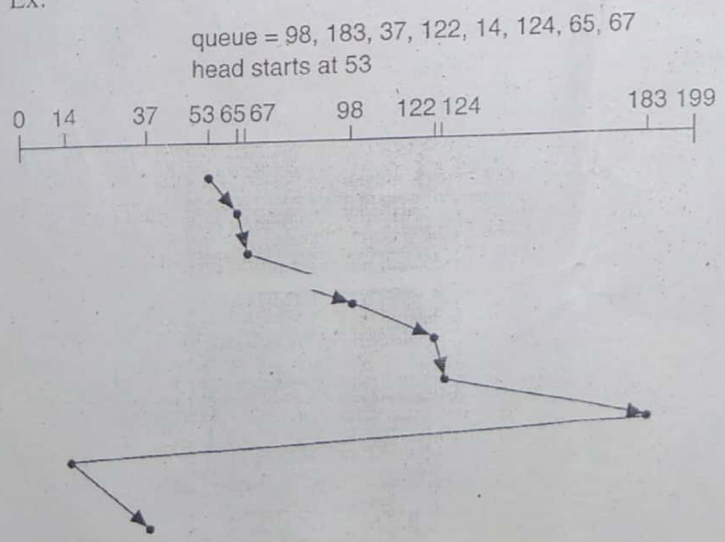


5) C-LOOK Scheduling:

a) This C-LOOK is the version of C-SCAN

b) More commonly, the arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk.

Ex:


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199