Unit - V

File System Interface and operations:

- Access methods
- Directory structure
- Protection
- File system structure
- Allocation Methods
- Free space management
- Usage of open Create, read, write close, lseek, stat, ioctl system calls.

operations:

ad, write
n calls.

File System Management

File : A file can be defined as a data structure which stores the sequence of records.

Files are stored in a file system, which may exist on a disk or in the main memory.

File can be simple (plain text) or complex (specially formatted).

The collection of files is known as Directory. The collection of directories at the different level is known as File system.

Attributes of the file:

1. Name :

Every file carries a name by which the file is recognized in the file system. One directory cannot have two files with the same name.

2. Identifier :

Along with the name. Each file has its own extension which identifies the type of the file. For example, a text file has the extension .txt, A video file can have the extension .mp4.

3. Type:

In a file system, the files are classified in different types such as video files, audio files, text files, executable files.

4. Location:

In the file system, there are several locations on which, the files can be stored. Each file carries its location as a its attribute.

5. Size:

The size of the file is one of its most important attribute. By size of the file, we will know the no. of bytes acquired by the file in the memory.

6. Protection:

The Admin of the computer may want the different protections for the different files. Therefore each file carries its own set of permissions to the different group of users.

7. Date

Eve
stamp
date o
modifie

Operat

The
Can be
1. Cr
C
import
types o
metho
used
proce
file
crea
2.

Crea
Write
point
wh

7. Date and Time:

Every file carries a time stamp which contains the time and date on which the file is last modified.

Operations on the file :-

There are various operations which can be implemented on a file.

1. Create:

Creation of the file is the most important operation on the file. Different types of files are created by different methods for example text editors are used to create a text file, word processors are used to create a word file and image editors are used to create the image file.

2. Write:

Writing the file is different from Creating the file. The OS maintains a write pointer for every file which points to the position in the file from which, the needs to be written.

3. Read:

Every file is opened in three modes: Read, write and append. A Read pointer is maintained by the OS, pointing to the position up to which the data has been read.

4. Re-position:

Repositioning is simply moving the file pointer forward or backward depending upon the user's requirement. It is also called as seeking.

5. Delete:

Deleting the file will not only delete all the data stored inside the file, It also deletes all the attributes of the file. The space which is allocated to the file will now become available and can be allocated to the other files.

6. Truncate:

Truncating is simply deleting the file except deleting attributes. The file is not Completely deleted

although the
inside the fi

I File Access

1. Sequenti
2. Direct
3. Indexe

files sto
used, this in
and read into
information
in several

1. Sequen

The
Sequential
file is, p
after the
is by far
example, co
access fil

In
read the
is maintai
the base

although the information stored inside the file get replaced.

File Access methods:

1. Sequential Access
2. Direct Access
3. Indexed Access.

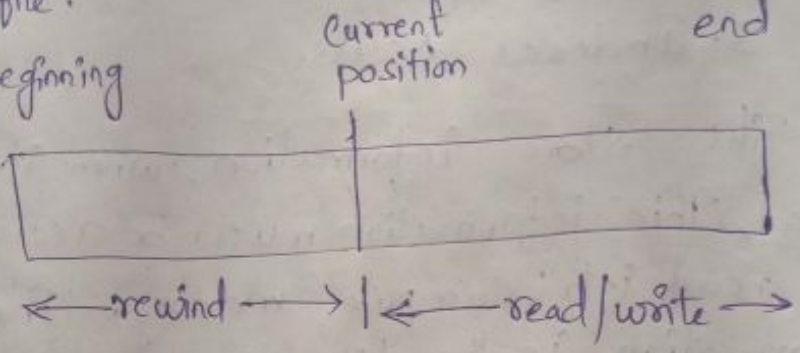
Files store information, when it is used, this information must be accessed and read into computer memory: The information in the file can be accessed in several ways.

1. Sequential access:-

The simplest access method is Sequential access. Information in the file is processed in order, one record after the other. This mode of access is by far the most common; for example, editors and compilers usually access files in this fashion.

In sequential access, the OS read the file word by word. A pointer is maintained which initially points to the base address of the file.

If the user wants to read first word of the file then the pointer provides that word to the user and increases its value by word. This process continues till the end of the file.



Modern word systems do provide the concept of direct access and indexed access but the most used method is sequential access due to fact that most of the files such as text files, audio files, video files, etc need to be sequentially accessed.

Direct Access :-

Another method is direct access (or relative access). A file is made up of fixed length logical records that allow programs to read and write

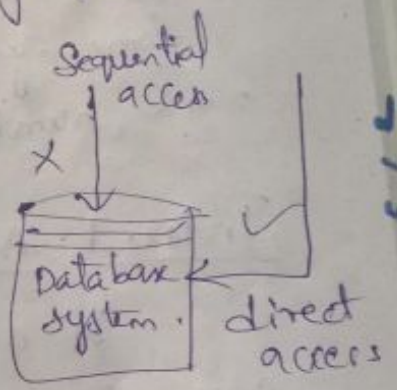
records rapidly in no particular order.

The direct access method is based on a disk model of a file, since disk allow random access to any file block.

For, direct access, the file is viewed as a numbered sequence of blocks or records.

Thus, we may read block 14, then read block 53 and then write block 7. There are no restrictions on the order of reading and or writing for a direct access file.

Sequential access	implementation for direct access
reset	CP = 0;
read next	read CP; CP = CP + 1;
write next	write CP; CP = CP + 1;



Direct access files are of great use for immediate access to large amount of information.

The file operations must include the be modified to include the block number as a parameter.

The block number provided by the user to the operating system is normally a relative block number.

A relative block number is an index relative to the beginning of the file.

The use of relative block numbers allows the OS to decide where the file should be placed and helps to prevent the user from accessing portions of the file system that may not be part of her file.

Some systems starts their relative block numbers at 0. others at 1.

Indexed Access :-

If a file can be sorted on any of the filed then an index can be assigned to a group of certain records. However, A particular record can be accessed by its index. The index is nothing but the address of a record in the file.

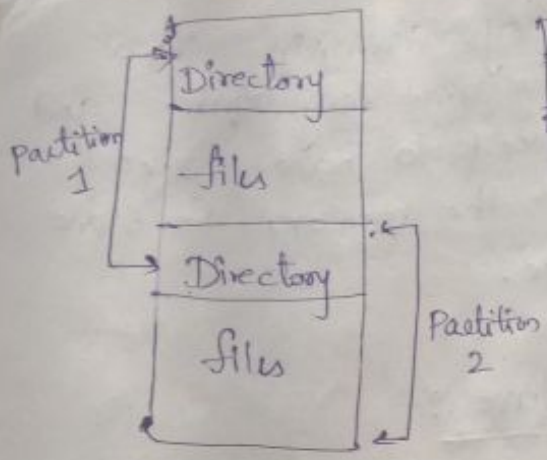
In index accessing, searching in a large database became very quick and easy but we need to have some extra space in the memory to store the index value.

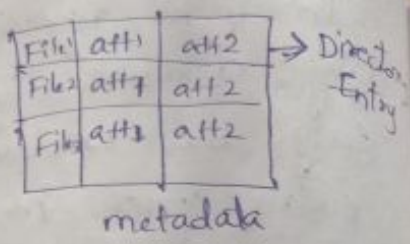
Directory structure :-

Directory can be defined as the listing of the related files on the disk. The directory may store some or the entire file attributes.

To get the benefit of different file systems on different O.S. A hard disk can be divided into the no. of partitions of diff size. The partitions are also called volumes or mini disks.

Each partition must have at least one directory. In which, all the files of the partition can be listed. A directory entry is maintained for each file in the directory which stores all the info related to that file.





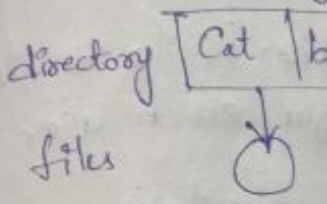
A directory can be viewed as a file which contains the metadata of the bunch of the files.

Every directory supports a no. of common operations on the file.

1. File creation
2. Search for the file.
3. File deletion.
4. Renaming the file.
5. Traversing file
6. Listing of files.

Single level D

The sim
is the single
are contained
which is easy

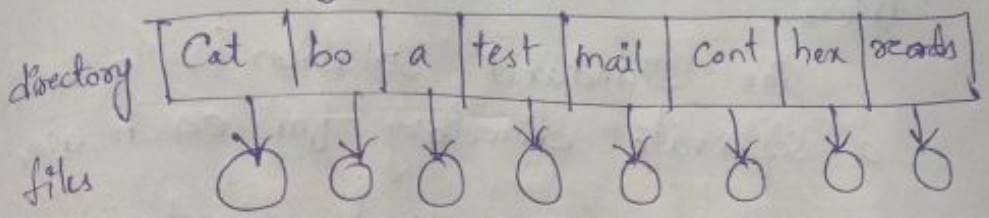


A single
limitations,
of files
system ha
since, a
directory,

Eve
level
difficu
of all
files incr
for a u
files on
equal
another
many

Single level Directory :-

The simplest directory structure is the single level directory. All files are contained in the same directory which is easy to support and understand.



A single level directory has significant limitations, however, when the number of files increases or when the system has more than one user. Since, all files are in the same directory, they must have unique names.

Even a single user on a single level directory may find it difficult to remember the names of all the files as the number of files increases. It is not uncommon for a user to have hundreds of files on one computer system and an equal no. of additional files on another system. Keeping track of so many files is a daunting task.

Two level directory :-

As we know, a single level directory often leads to confusion of file names among different users.

The standard solution is to create a separate directory for each user.

In the two level directory structure, each user has its own user file directory (UFD). The UFDs have similar structures, but each lists only the files of a single user.

When a user job starts or a user logs in, the system's master file directory (MFD) is searched.

The MFD is indexed by user name or account number, and each points to the UFD for that user.

When a user refers to a particular file, only his own UFD is searched. Thus diff users may have files with the same name, as long as all the file names within each UFD are unique.

To create

as searches

ascertain

that name

To delet

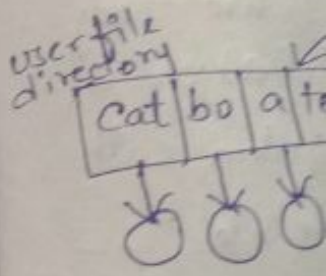
its search

it cannot

user's file

Master

direct



Altho

solves th

still h

effectiv

Isolation

users

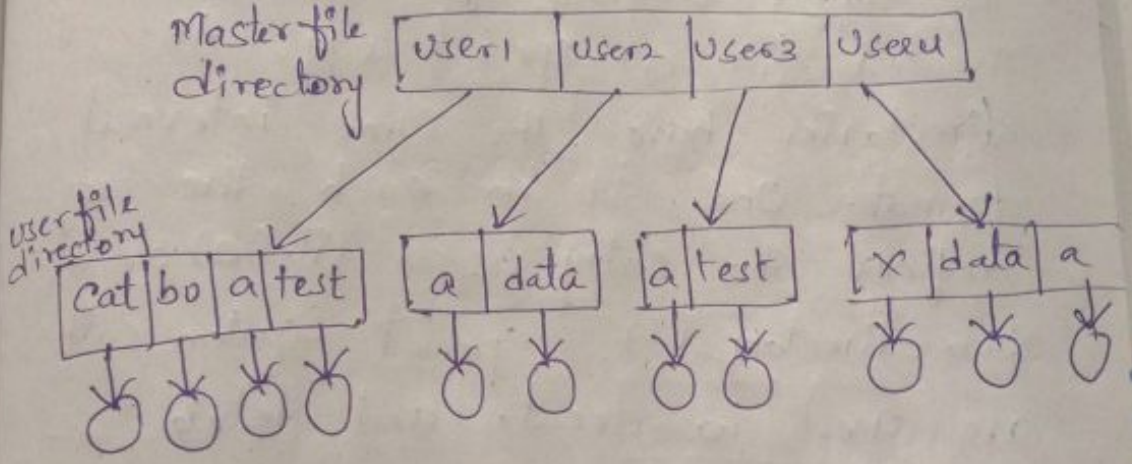
disadv

to coo

to ac

To create a file for a user, the OS searches only that user's UFD to ascertain whether another file of that name exists.

To delete a file, the OS confines its search to the local UFD; thus, it cannot accidentally delete another user's file that has same name.



Although two level directory structure solves the name collision problem, it still has disadvantages. This structure effectively isolates one user from another. Isolation is an advantage when the users are completely independent but disadvantage is when the users want to cooperate on some task and to access one another's files.

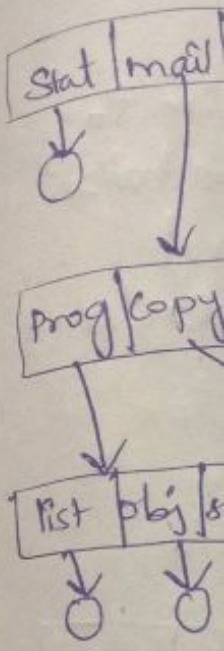
Tree structured directory :-

A tree is the most common directory structure. The tree has a root directory, and every file in the system has a unique path name.

A directory contains a set of files or subdirectories. A directory is simply another file, but it is treated in a special way. All directories have the same internal format. One bit in each directory defines the entry as a file (0) or a subdirectory (1). Special system calls are used to create and delete directories.

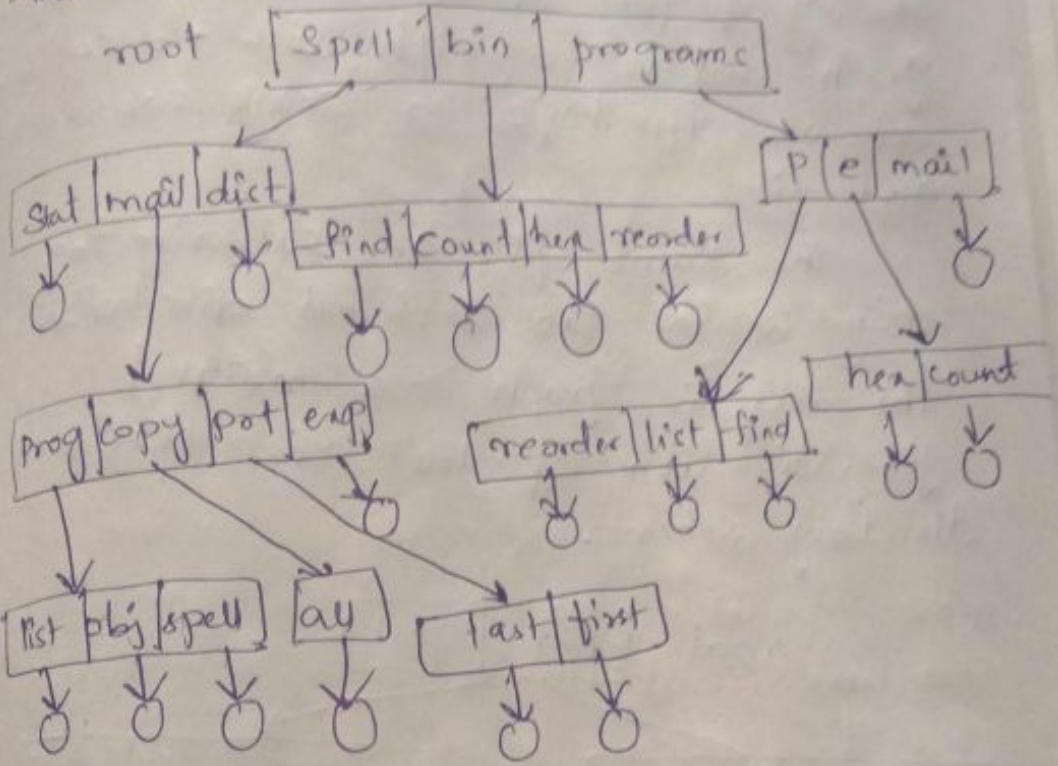
In normal use, each process has a current directory. The current directory should contain most of the files that are of current interest to the process. When reference is made to a file, the current directory is searched. If a file is needed that is not in the current directory, then the user usually must either

Specify
current
that fil
root



Path
absolu
name
down
the
relati
curr

Specify a pathname or change the current directory to be directory holding that file.

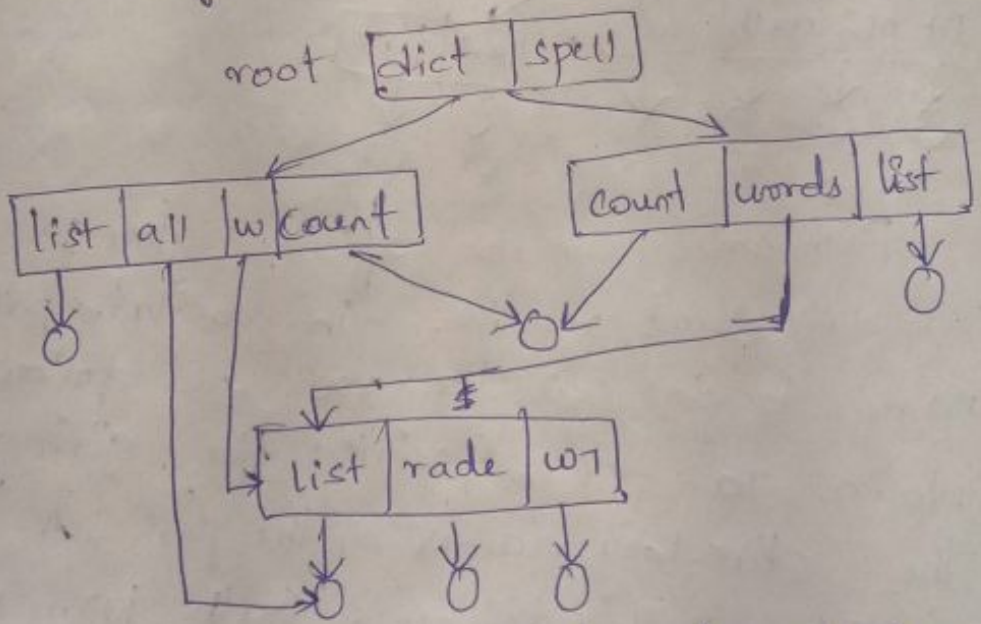


Pathnames can be of two types : absolute and relative. An absolute path name begins at the root and follows down to the specified file, giving the directory names on the path. A relative path defines a path from the current directory.

Acyclic graph Directory :-

A tree structure prohibits the sharing of files or directories. An acyclic graph that is a graph with no cycles allows directories to share subdirectories and files.

The same file or subdirectory may be in two different directories. The acyclic graph is a natural generalization of the tree structured directory scheme.



It is important to note that a shared file is not the same as two copies of the file. With two copies, each programmer can view the copy rather than the original, but if one programmer changes the file,

the changes will not appear in the other's copy.

With a shared file, only one actual file exists, so any changes made by one person are immediately visible to the other. sharing is particularly important for subdirectories a new file created by one person will automatically appeared in all the shared subdirectories.

Shared files and subdirectories can be implemented in several ways:

A common way is to create a new directory entry called a link.

A link is effectively a pointer to another file or subdirectory.

we resolve the link by using the path name to locate the real file.

Links are easily identified by their format in the directory entry. and are effectively named indirect pointers.

The OS ignores those links when traversing directory trees to preserve the acyclic structure of the system.

Another common approach to implementing shared files is simply to duplicate all information about them in both sharing directories.

Thus both ~~director~~ entries are identical and equal.

A major problem with duplicate directory entries is maintaining consistency when a file is modified.

An acyclic directory structure is more flexible than a simple tree structure, but it is also more complex. Several problems must be considered carefully. A file may now have multiple absolute path names. Consequently distinct file names may refer to the same file.

III Protection

When i
Computer sys
safe from
and improp

Reliabili
by duplica
computers
automatica
to tape
maintain
System b

file s
H/w prob
reading
failures,
extremes
may be
in the
cause

Pr
ways
might
Prohibiti

Protection :-

When information is stored in a computer system, we want to keep it safe from physical damage (reliability) and improper access (protection).

Reliability is generally provided by duplicate copies of files. Many computers have system programs that automatically copy disk file files to tape at regular intervals. to maintain a copy should a file system be accidentally destroyed.

File systems can be damaged by H/w problems such as error in reading or writing, power surges or failures, head crashes, dirt, temperature extremes and ~~bad~~ vandalism. files may be deleted accidentally. Bugs in the file system S/w can also cause file contents to be lost.

Protection can be provided in many ways. For a single user system, we might provide ~~complete~~ protection by ~~prohibiting access~~.

physically removing the floppy disk and locking them in a desk drawer or file cabinet. In a multi user system, however, other mechanisms are needed.

Types of Access :-

The need to protect files is a direct result of the ability to access files.

systems that do not permit access to the files of other users do not need protection.

Thus, we could provide complete protection by prohibiting access. Alternatively we could provide free access with no protection.

Both approaches are too extreme for general use. what is needed is controlled access.

Protection mechanisms provide controlled access by limiting the types of file access that can be made. Access is permitted or denied depending on several factors, one of which the type of access requested.

several different types of operations may be controlled:

Read: Read from the file.

write: write or rewrite the file.

execute: load the file into memory and execute it.

Append: write new information at the end of the file.

Delete: Delete the file and free its space for possible reuse.

List: list the name and attributes of the file.

Other operations, such as renaming, copying and editing the file, may also be controlled.

Access control :-

The most common approach to the protection problem is to make access dependent on the identity of the user.

Different users may need different types of access to a file or directory.

The most general scheme to implement identity dependent access.

is to associate with each file and directory an access control list (ACL) specifying user names and the types of access allowed for each user.

When a user request access to a particular file, the OS checks the access list associated with that file. If that user is listed for the requested access, the access is allowed. Otherwise, a protection violation occurs, and the user job is denied access to the file.

This approach has the advantage of enabling complex access methodologies.

The main problem with access list is their length. If we want to allow everyone to read a file, we must list all users with read access. This techniques has two undesirable consequences:

* Constructing such a list may be tedious and unrewarding task, especially if we do not know in advance the list of users in the system.

* The directo
of fixed size
variable size
complicated

These pr
by use of
access list.

To con
access cont
recognize
users in

Owner:
file is

Group
sharing
similar
work gro

Unive
Systen

* The directory entry, previously of fixed size, now needs to be of variable size, resulting in more complicated space management.

These problems can be resolved by use of a condensed version of the access list.

To condense the length of the access control list, many systems recognize three classifications of users in connection with each file

Owner: The user who created the file is the owner.

Group: A set of users who are sharing the file and need similar access is a group or work group.

Universe: All other users in the system constitute the universe.

File System structure :-

Disks provide the bulk of secondary storage on which a file system is maintained.

They have two characteristics that make them a convenient medium for storing multiple files:

1. A disk can be rewritten in place; It is possible to read a block from the disk, modify the disk/block, and write it back into the same place.

2. A disk can access directly any given block of information it contains.

To provide efficient and convenient access to the disk, the OS imposes one or more file systems to allow the data to be stored, located and retrieved easily.

A file system poses two quite different design problems.

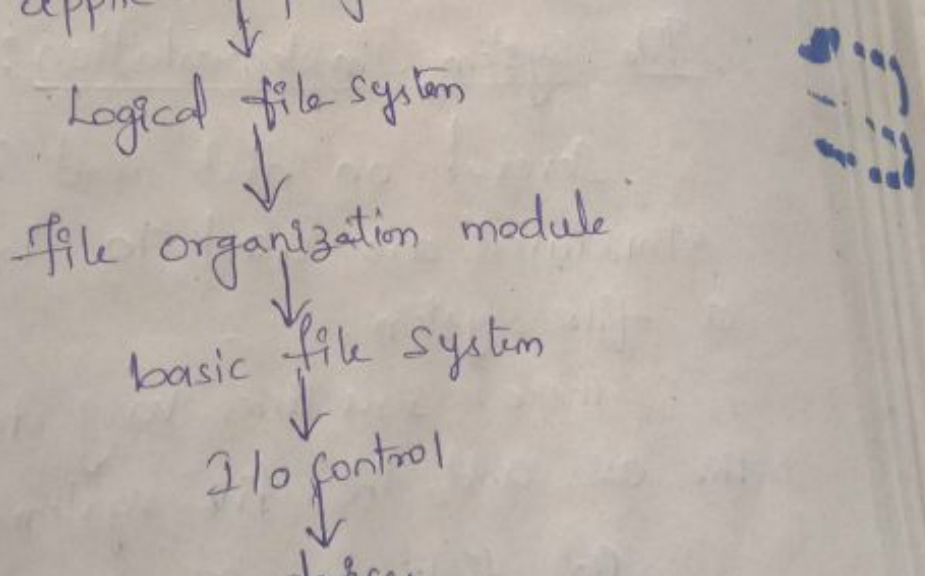
1. The first problem is defining

how the file system should look to the user.

2. The second problem is creating algorithms and data structures to map the logical file system on to the physical secondary storage.

The file system itself is generally composed of many different levels.

The basic file system needs only to issue generic commands to the appropriate device driver to read and write physical blocks on the disk.



Layered file system

The lowest level I/O control, consists of device drivers and interrupt handlers to transfer info between the main memory and disk system.

The file organization module knows about files and their logical blocks as well as physical blocks.

The logical file system manages the metadata information. Metadata includes all of the file system structure except the actual data.

A file-control block (FCB) contains info about the file, including ownership, permissions and location of the file contents.

File system Implementation :-

Several on disk and in memory structures are used to implement a file system.

These structures vary depending on the OS and the file systems.

On disk, the file system may contain info about how to boot an OS stored there, the total number of blocks, the number and location of free blocks, directory structure and individual files.

A boot control block: can contain information needed by the system to boot an OS from that volume

If the disk does not contain OS, this block can be empty. It is typically the first block of volume.

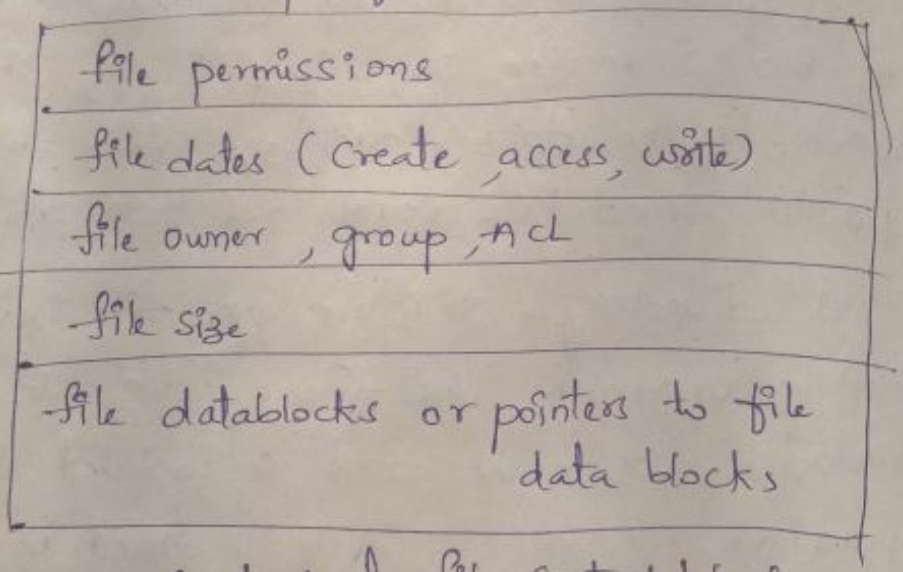
A volume control block contains volume (or partition) details, such as the number of blocks in the partition, size of the blocks, free block count and free-block pointers, and FCB count and FCB pointers.

A directory structure per file system is used to organize the files. In UFS this includes file names and associated inode numbers. In NTFS it is stored in the master file table.

A boot control block called boot block in UFS. In NTFS it is the partition boot sector.

A In UFS volume control block called a Super block. In NTFS, it is stored in the master file table.

As per file FCB (file control block) contain many details about the file, including file permissions, ownership, size and locat of the data blocks. In UFS, this is called the inode. In NTFS, this information is actually stored within the master file table, which uses a relational database structure, with a row per file.

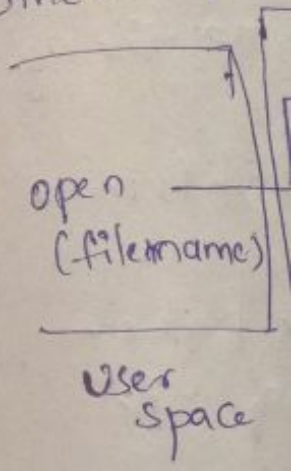


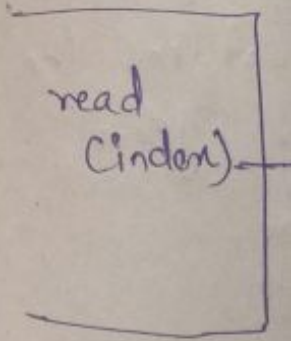
A typical file control block.

The in-memory information is used for both file system management and performance improvement via caching. The data are loaded at mount time and discarded at dismount.

The An in memory table contains information about each mounted volume.

An in-memory
holds the direc
accessed direct
The system
a copy of th
as well as
The per-
a pointer to
system wide
other inform



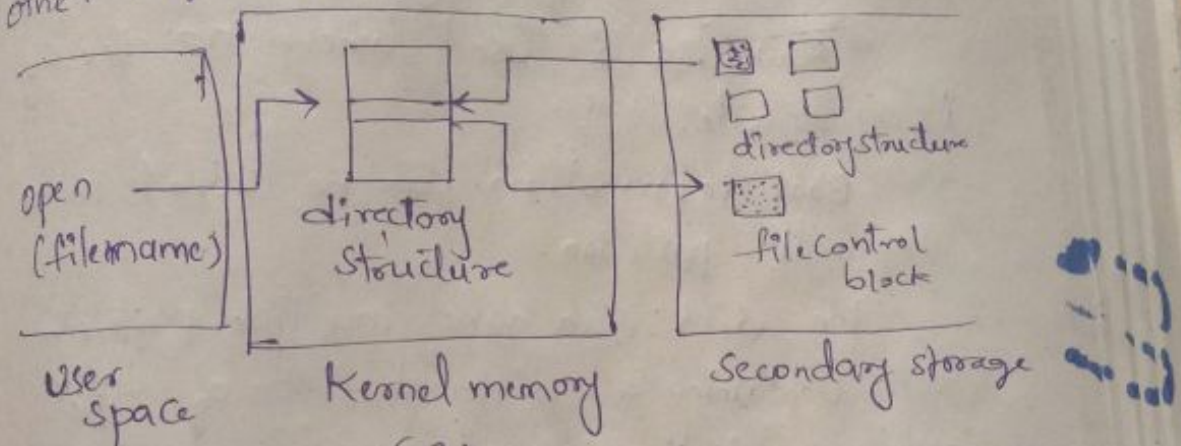


In
Ca

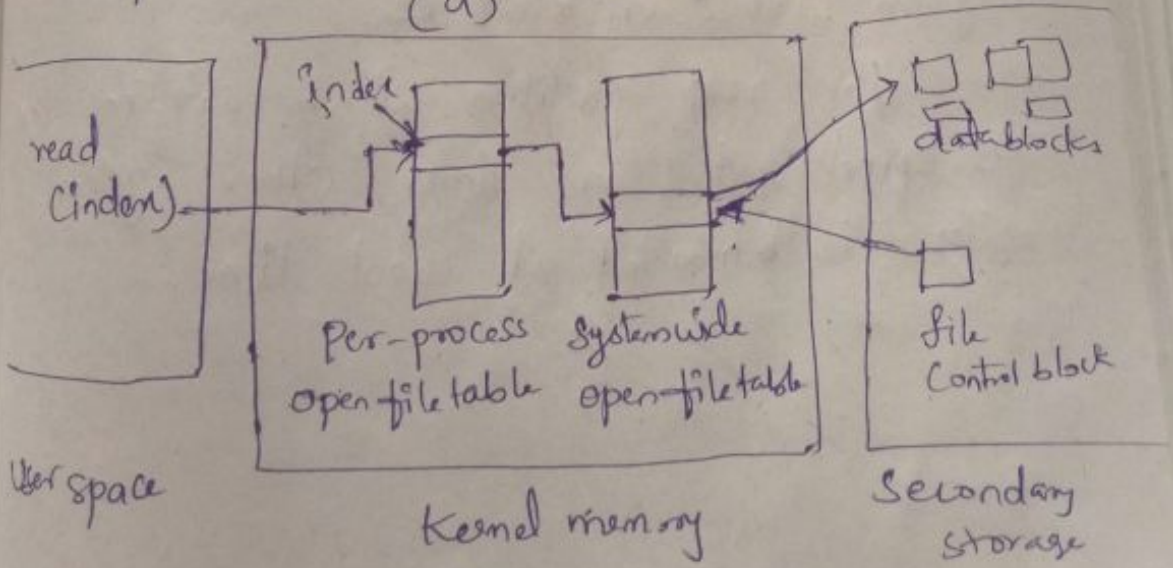
An in-memory directory structure caches the directory information of recently accessed directories.

The system-wide open file table contains a copy of the FCB of each open file, as well as other information.

The per-process open file table contains a pointer to the appropriate entry in the system wide open file table as well as other information.



(a)



(b)

In memory file system structures (a) file open (b) file read.

Partitions and mounting :-

The layout of a disk can have many variations, depending on the OS. A disk can be sliced into multiple partitions, or a volume can span multiple partitions on multiple disks.

Each partition can be either "raw", containing no file system, or "cooked", containing a file system.

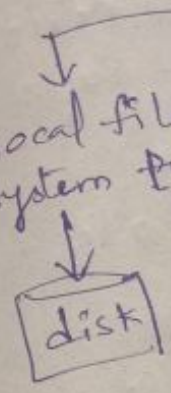
Raw disk is used where no file system is appropriate.

Boot information can be stored in a separate partition.

The disk can have multiple partitions, each containing a different type of file system and a different OS.

The root partition which contains the OS kernel and some times other system files, is mounted at boot time.

Virtual file



Sch

Th

interfa

write

T

Syste

funct

1)

oper

def

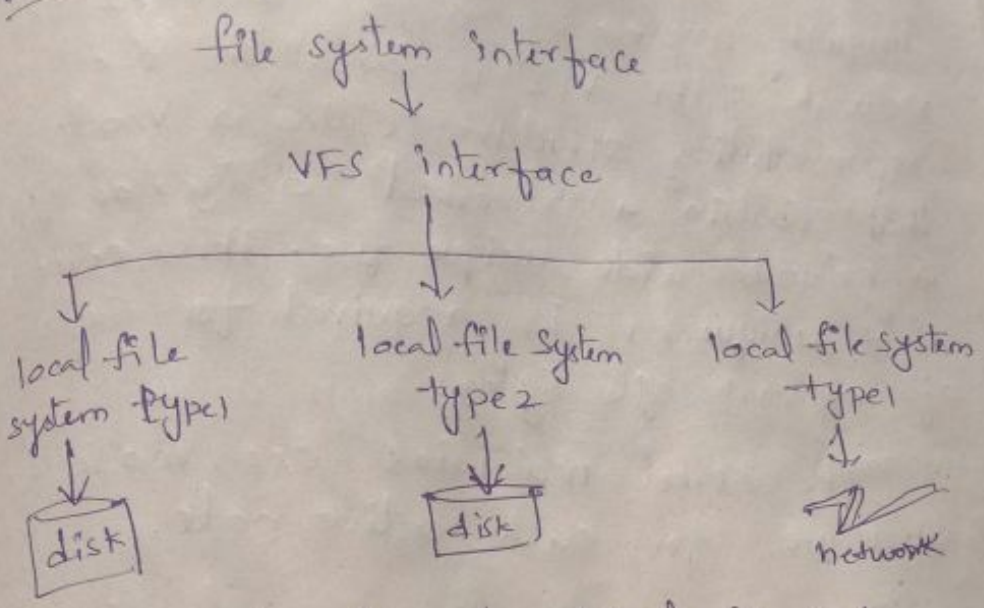
im

Co

th

of

Virtual file systems :-



Schematic view of virtual file system.

The first layer is the file system interface, based on the open(), read(), write() and close() calls on file descriptors.

The second layer is the virtual file system layer. It serves two important functions.

1) It separates file system generic operations from their implementation by defining a clean VFS interface. Several implementations for the VFS interface may coexist on the same machine, allowing transparent access to different types of file systems mounted locally.

2. The VFS provides a mechanism for uniquely representing a file throughout a network. The VFS is based on file representation structure, called a vnode, that contain a numerical designator for a network-wide unique file. This network wide uniqueness is required for support of network file systems.

The kernel maintains one vnode structure for each active node.

Thus, VFS distinguishes local files from remote ones, and local files are further distinguished according to their file system types.

6. Directory Implementation:-

The selection of implementing a directory is to use a linear list of file names with pointers to the data blocks. This method is simple to program but time consuming to execute.

Directory Implementation :-

The selection of directory allocation and directory management algorithms significantly affects the efficiency, performance and reliability of the file system.

Linear list :-

The simplest method of implementing a directory is to use a linear list of file names with pointers to the data blocks. This method is simple to program but time consuming to execute.

To create a new file, we must first search the directory be sure that no existing file has the same name. Then, we add a new entry at the end of the directory.

To delete a file, we search the directory for the named file, then the release the space allocated to it.

The real disadvantage of a linear list of directory entries is that finding a file requires a linear search. Directory information is used frequently, and users will notice if access to it is slow.

Hash table :-

Another data structure used for a file directory is a hash table. With this method a linear list stores the directory entries, but a hash data structure is also used.

The hash table takes a value computed from the file name and returns a pointer to the file name in the linear list.

Therefore it can greatly decrease the directory search time.

Insertions and deletion are also fairly straightforward although some provision must be made for collisions - situations in which two files names hash to the same location.

The major difficulties with a hash table are its generally fixed size and the dependence of the hash function on that size.

Allocation met

There ar

be used to

files. selection

method will

performance

Allocation m

which the

files will

the --

disk spa

1. Co

2. I

3. I

1. Contig

If

file

logical

such

Contig

Blocks

Allocation methods:-

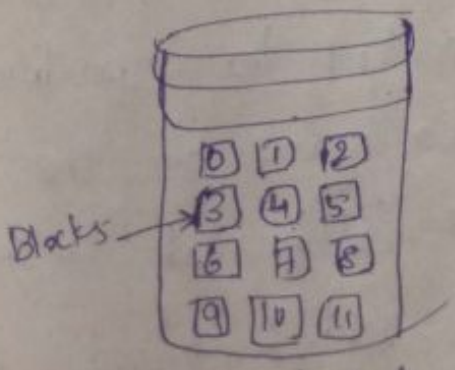
There are various methods which can be used to allocate disk space to the files. Selection of an appropriate allocation method will significantly affect the performance and efficiency of the system. Allocation method provides a way in which the disk will be utilized and the files will be accessed.

The three major methods of allocating disk space are:

1. Contiguous Allocation Method.
2. Linked Allocation Method.
3. Indexed Allocation Method.

1. Contiguous Allocation:-

If the blocks are allocated to the file in such a way that all the logical blocks in the hard disk then such allocation scheme known as Contiguous allocation.



Hard disk

filename	start	length	Allocated blocks
abc	0	3	0,1,2
x	4	2	4,5
y	9	3	9,10,

Directory.

There are three files in the directory. The starting block and the length of each file are mentioned in the table. We can check in the table that the contiguous blocks are assigned to each file as per its need.

Advantages :-

1. It is simple to implement.
2. We will get excellent read performance.
3. Supports random access into files.

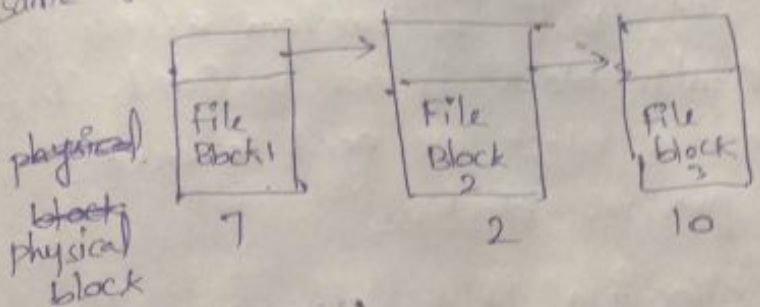
Disadvantages :-

1. The disk will become fragmented.
2. It may be difficult to have a file grow.

Linked list Allocation :-

Linked list allocation solves all problems of contiguous allocation. In linked allocation, each file is considered as the linked list of disk blocks. However, the disks blocks allocated to a particular file need not to be contiguous on the disk. Each disk block allocated to a file contains a pointer which points

to the next disk block allocated the same file.



Advantages :-

There is no external fragmentation with linked allocation.

Any free block can be utilized in order to satisfy the file block requests.

File can grow continue to grow as long as the free blocks are available.

Directory entry will only contain the starting block address.

Disadvantages:

Random access is not provided.

pointer require some space in the disk blocks.

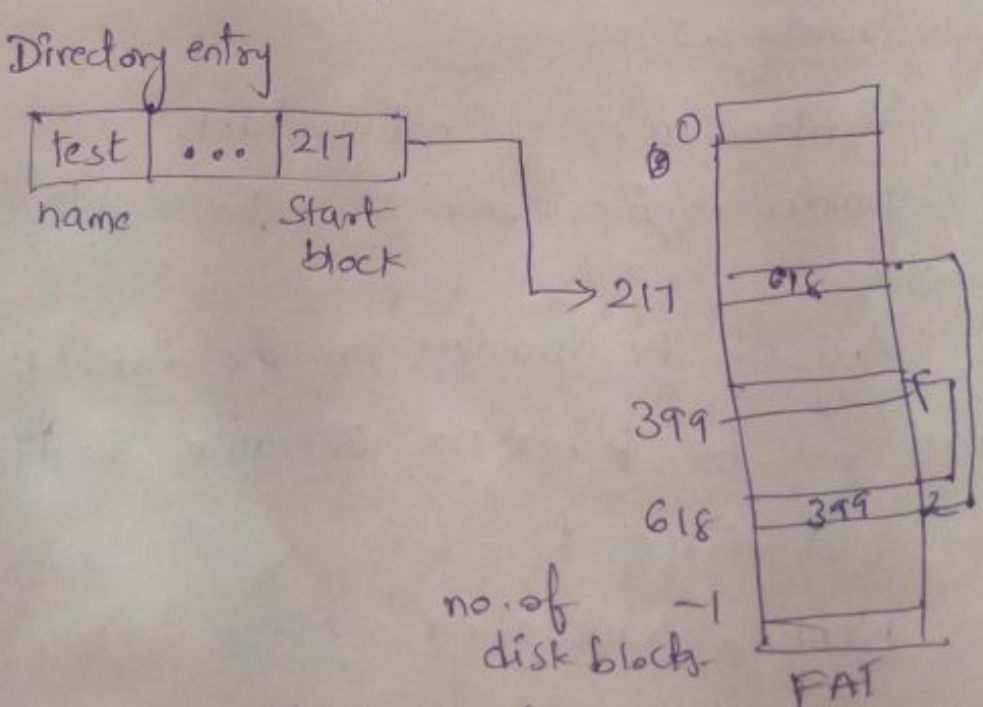
Any of the pointers in the linked list must not be broken otherwise the file will get corrupted.

Need to traverse each block.

The main disadvantage of linked list allocation is that the random access to a particular block is not provided. In order to access a block we need to access all its previous blocks.

File allocation table overcomes this drawback of linked list allocation.

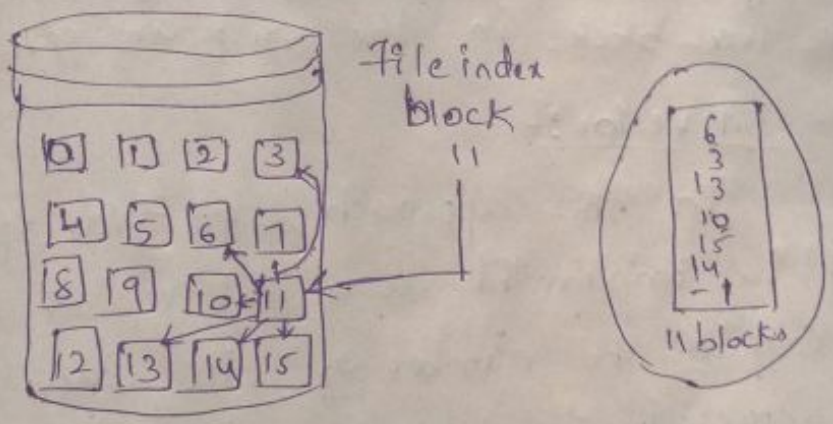
In this scheme file allocation table is maintained, which gathers all the disk block links. The table has one entry for each disk block and is indexed by block number. The table has one entry for each disk block and is indexed by block number. File allocation table needs to be cached in order to reduce the number of head seeks.



File Allocation table

Indexed Allocation :-

Instead of maintaining a FAT of all the disk pointers, Indexed allocation scheme stores all the disk pointers in one of the blocks called as indexed blocks. Indexed block doesn't hold the file data, but it holds the pointers to all the disk blocks allocated to that particular file. Directory entry will only contain the index block address.



Advantages :-

- Supports direct access
- A bad data block causes the lost of only that block.

Disadvantages :-

- A bad index block could cause the lost of entire file.
- Size of file depends upon the number of pointers, index block can hold

- Having a index block for a small file is totally wastage.
- More pointer overhead.

Free space management :-

A file system is responsible to allocate the free blocks to the file therefore it has to keep track of all the free blocks present in the disk. There are mainly two approaches by using which, the free blocks in the disk are managed.

1. Bit vector :-

In this approach, the free space list is implemented as a bit map vector. It contains the number of bits where each bit represents each block.

If the block is empty then the bit is 1. otherwise it is 0. Initially all the blocks are empty therefore each bit in the bit map vector contains 1.

2. Linked list :-

Another approach to free space management is to link together all the

free disk bl
the first free
on the disk
The file
to the next
on.

3. Grouping :

A mod
is to sto
in the firs
these bloc
block co
n free b
of a large
be found
when th
is used.

4. Coun

An
advanta
several
or freed
When

free disk blocks, keeping a pointer to the first free block in a special location on the disk and caching it in memory. The file block contains a pointer to the next free disk block and so on.

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 blocks are actually free. The last block contains the addresses of another n free blocks and so on. The addresses of a large number of free blocks can now be found quickly, unlike the situation when the standard linked list approach is used.

4. Counting :-

An another approach is to take advantage of the fact that, generally, several contiguous blocks may be allocated or freed simultaneously, particularly when space is allocated with the contiguous

allocation algorithm or through clustering. Thus, rather than keeping a list of n free disk addresses, we can keep the address of the first free block and the number n of free contiguous blocks that follow the first block.

Disk Scheduling Algorithms :-

As we know, a process needs two types of time, CPU time and I/O time. For I/O, it requests the OS to access the disk.

However, the OS system must be fare enough to satisfy each request and at the same time, OS must maintain the efficiency and speed of process execution.

The technique that OS uses to determine the request which is to be satisfied next is called disk scheduling.

Terms :-

Seek time: Time taken in locating the disk arm to a specified track where the read/write request will be satisfied

clustering.
list of
n keep the
lock and
guous
st block.

needs two
time.
access

st be
equest and
naintain
cess

to
to be
cheduling.

ating
ack
satisfied.

Rotational latency:

Time taken by the desired sector to rotate itself to the position from where it can access the R/W heads.

Transfer time:

Time taken to transfer the data.

Disk access time:

Disk access time is given as

Disk access time = Rotational latency + Seek time + Transfer time.

Disk response time:-

The average time spent by each request waiting for I/O operation.

Purpose of Disk scheduling:

To select a disk request from the queue of I/O request and decide the Schedule when this request will be Processed.

Goals:-

Fairness
High throughput
minimal traveling head time.

Disk scheduling Algorithms :-

1. First come First Serve (FCFS) Scheduling Algorithm
2. Shortest seek time first (SSTF) algorithm
3. SCAN Scheduling Algorithm.
4. C-SCAN Scheduling Algorithm
5. LOOK Scheduling Algorithm
6. C-LOOK Scheduling Algorithm.

1. First come first Serve (FCFS) Scheduling Algorithm :-

It is simplest disk scheduling algorithm. It serves the I/O request in the order in which they arrive. There is no starvation in this algorithm, every request is saved.

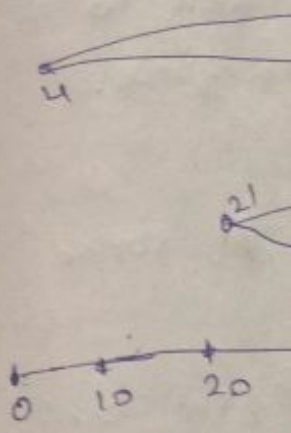
Disadvantages :

- The scheme does not optimize the seek time
- The request may come from different processes therefore there is possibility of inappropriate movement of the head.

Example :-

Consider the f
for a disk with
50, 89, 52, 61, 87,

Header point
directions. Fin
in cylinders usi

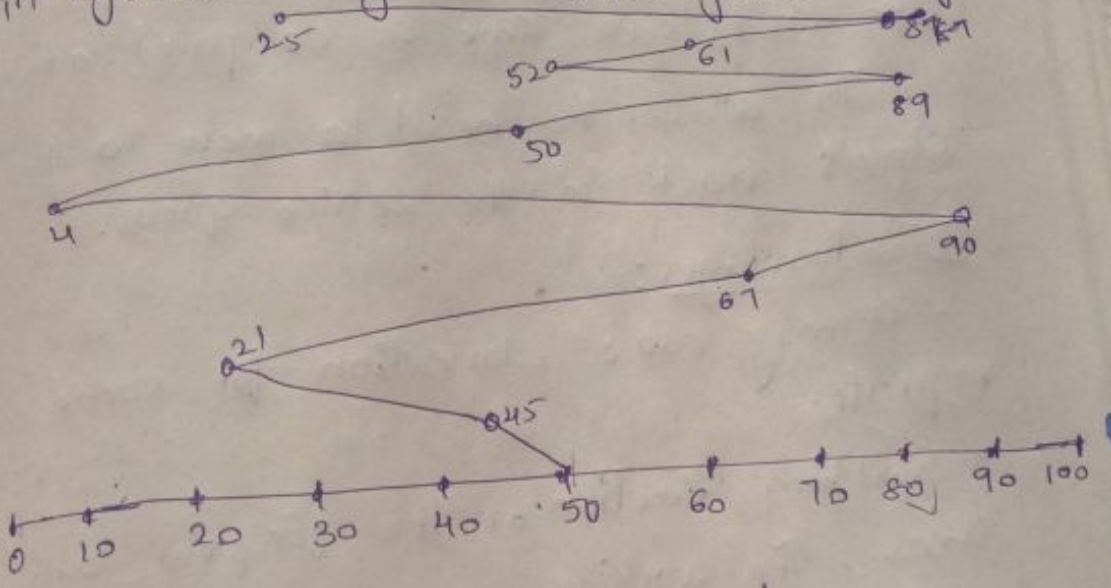


No. of cy
= (50
(90-4)
+ (87-
= 5
= 3

Example:-

Consider the following disk request sequence for a disk with 100 tracks 45, 21, 67, 90, 4, 50, 89, 52, 61, 87, 25.

Header point starting at 50 moving in left directions. Find the no. of heads movements in cylinders using FCFS ~~Scheduling~~ Scheduling?



No. of cylinders moved by head

= (50-45) + (45-21) + (67-21) + (90-67) + (90-4) + (50-4) + 89-50) + (61-52) + (87-61) + (87-25)

= 5+24+46+23+86+46+49+9+26+62

= 376.

2. Shortest seek time first (SSTF) Schedul...

Algorithm :-

SSTF algorithm selects the disk I/O request which requires the least disk arm movement from its current position regardless of the direction. It reduces the total seek time as compared to FCFS.

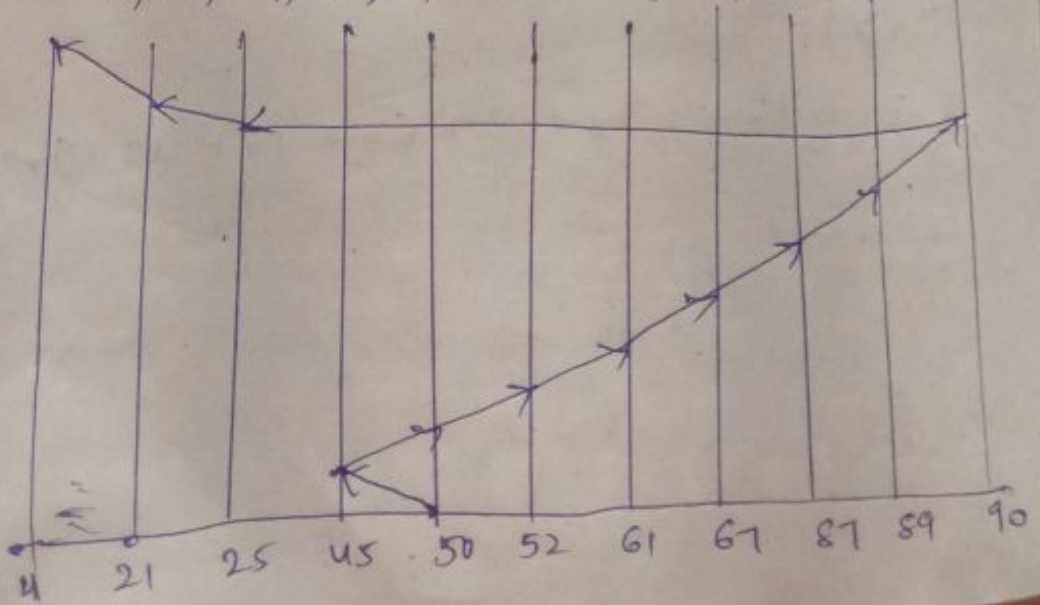
. It allows the head to move to the closest track in the service queue.

Disadvantages :-

- It may cause starvation for some requests.
- Switching direction on the frequent basis slows the working of algorithm.
- It is not the most optimal algorithm.

Example :- Head pointer = 50. no. of head movement =?

45, 21, 67, 90, 4, 89, 52, 61, 87, 25.



No. of Head
= 5+7+9+
= 136.

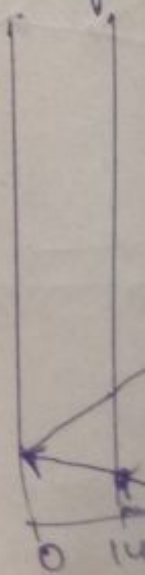
3. Scan Algorith

It is al

In this algo
particular
all the req
then it t
reverse di
in its path.

It u
elevator
the last
turns ba

Exampl
Head
no. of h



No. of Head movements in cylinders:

$= 5+7+9+6+20+2+1+65+4+17$

$= 136.$

3. Scan Algorithm :-

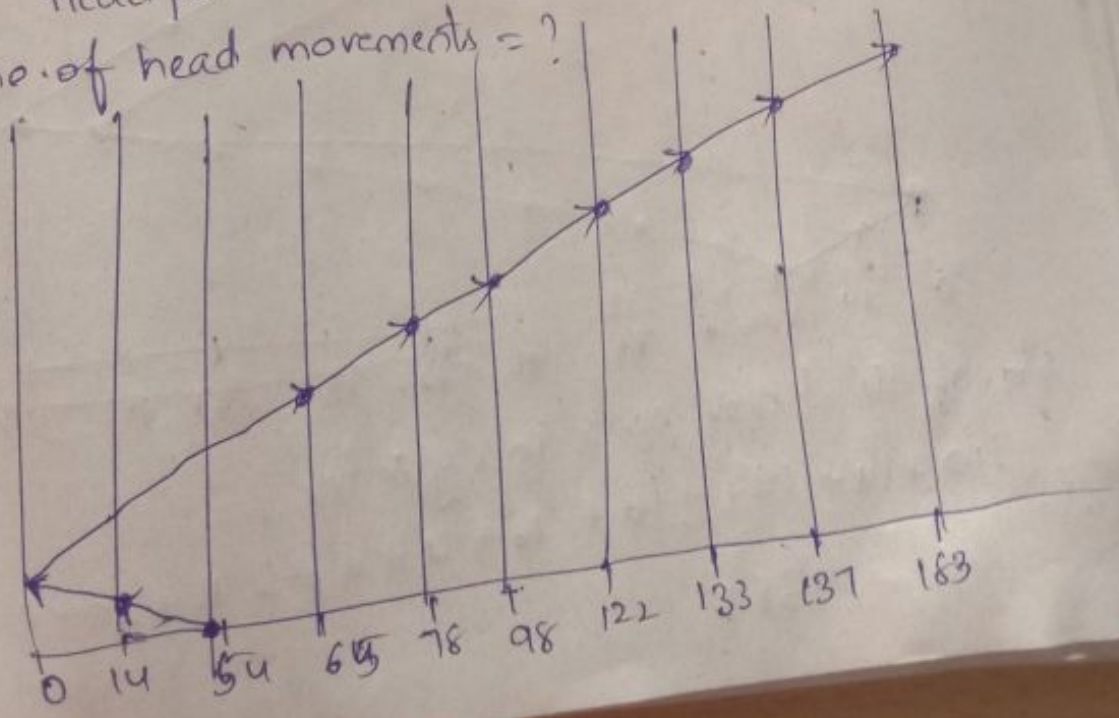
It is also called as Elevator-Algorithm. In this algorithm, the disk arm moves into a particular direction till the end, satisfying all the requests coming in its path, and then it turns back and moves in the reverse direction satisfying requests coming in its path.

It works in the way an elevator works, elevator moves in a direction completely till the last floor of that direction and then turns back.

Example :- 98, 137, 122, 183, 14, 133, 65, 78.

Head pointer = 54 left direction (→) movement

no. of head movements = ?



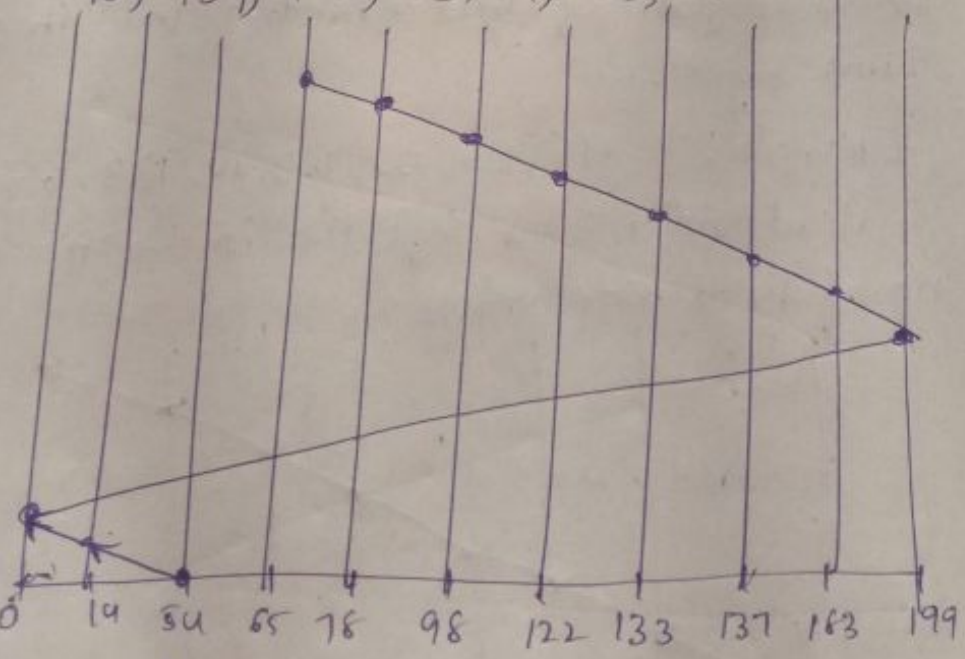
No. of head movements in cylinder =
40+14+65+13+20+24+11+4+46 = 237

C-SCAN Algorithm :-

In C-SCAN Algorithm, the arm of the disk moves in a particular direction servicing requests until it reaches the last cylinder, then it jumps to the last cylinder of the opposite direction without servicing any request then it turns back and start moving in that direction servicing the remaining requests.

Example : Headpointer = 54 (Left direction)

98, 137, 122, 183, 14, 133, 65, 78



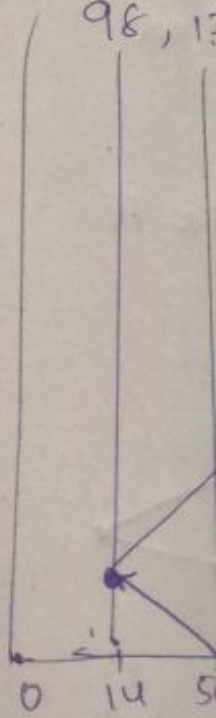
No. of Head movements in cylinder =
40+14+199+16+46+4+11+24+20+13
= 387

Look scheduling

It is like
to some extent
In this, the a
inwards (or out
in that direc
this algo
of SCAN Alg
move in one
of knowing
or not.

Example :-

98, 1



No. of he
= 40
= 20

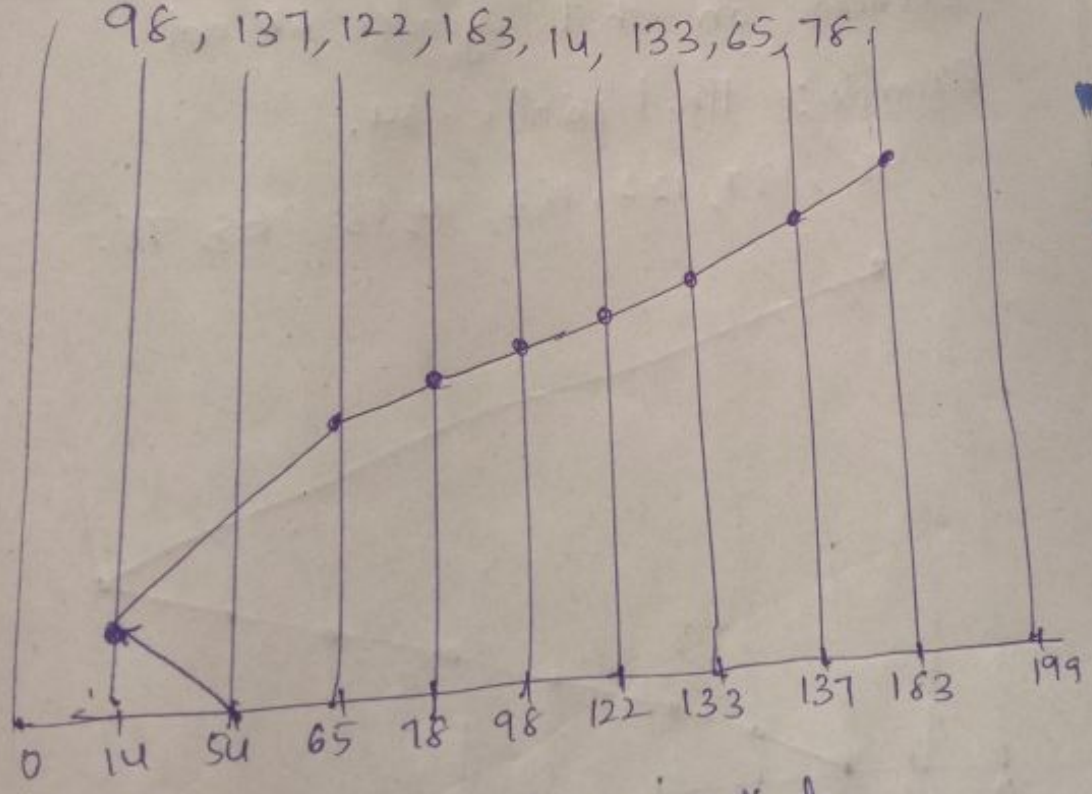
Look scheduling :-

It is like SCAN scheduling Algorithm to some extent the difference is that, In this, the arm of the disk stops moving inwards (or outwards) when no more request in that direction exist.

This algorithm overcomes the overhead of SCAN Algorithm which forces disk arm to move in one direction till the end regardless of knowing if any request exist in the direction or not.

Example :- Head pointer = 54

98, 137, 122, 183, 14, 133, 65, 78



No. of head movements in cylinder

= 40 + 51 + 13 + 20 + 24 + 11 + 4 + 46

= 209

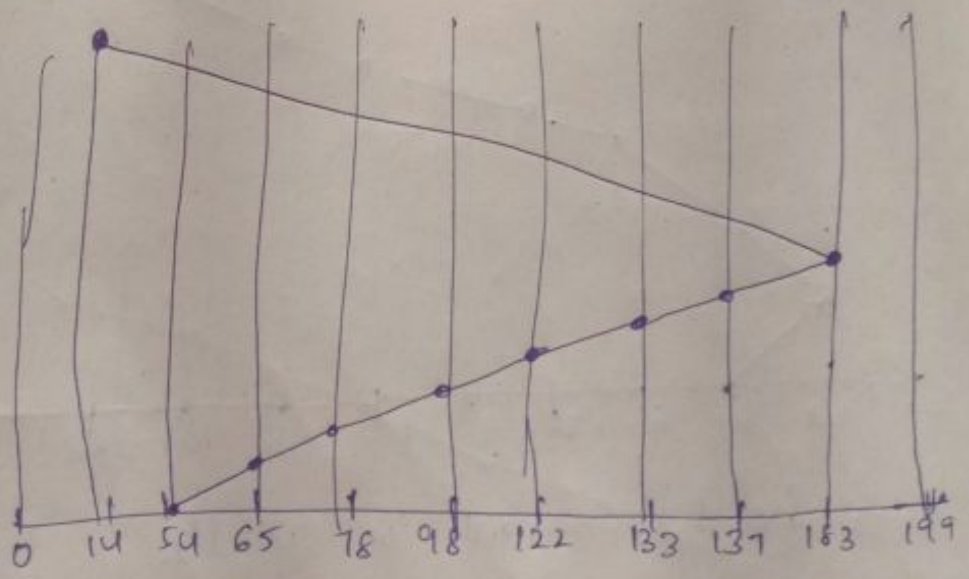
C-Look scheduling :-

C Similar to C-SCAN Algorithm to some extent. In this Algorithm, the arm of the disk moves outwards servicing requests until it reaches the highest request cylinder without servicing any request then it again start moving outwards servicing the remaining requests.

It is different from CSCAN algorithm in the sense that, CSCAN force the disk arm to move till the last cylinder regardless of knowing whether any request is to be serviced on that cylinder or not.

Example :- Head pointer = 54.

98, 137, 122, 183, 14, 133, 65, 78.



No. of Head movements in cylinder:

= 11+13+20+24+11+4+46+169

= 298