File system structure:

Ex Hard disks have two important properties that make them suitable for secondary storage of files in file systems.

→ Blocks of data can be rewritten in place, and they are directly accessing any block of data ✓

Disks offer the massive amount of secondary storage where a file system can be maintained. They have two characteristics which make them a suitable medium for storing various files

→ A disk can be used to rewrite in place, it is possible to read a chunk from the disk modify the chunk and write it back there in the same place.

→ A disk can access directly any given block of data it contains. Hence, it is easy to access any file either in sequence or at random and switching from one single file to another need only to move the read write head and wait for the disk to rotate to that specific location.

• Disks are usually accessed in physical blocks, rather than a byte at a time. Block sizes may range from 512 bytes to 4K or larger.

→ File system organize storage on disk drivers, and can be viewed as a layered design

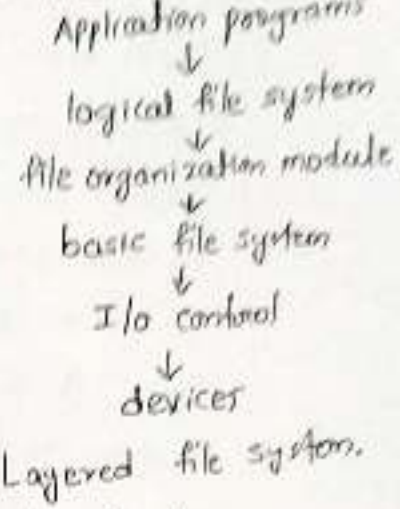
At the lower layer are the physical devices, consisting of the magnetic media, motors & controls, and the electronics connected to them and controlling them.

I/o control consist of device drivers, special software programs which communicate with the devices by reading and writing special codes directly to and from memory addresses corresponding to the controller cards register.

The basic file system level works directly with the device drivers in terms of retrieving and storing raw blocks of data without any consideration for what is in each block. Depending on the system, blocks may be referred to with a single block number, or with head sector cylinder combinations.

The file organization module knows about files and their logical blocks, and how they map to physical blocks on the disk. The logical file system deals with all of the meta data associated with a file.

The layered approach to file systems means that much of the code can be used in different file systems, and only certain layers need to be filesystem specific.



Layered file system.

File system implementation:

File systems store several important data structures on the disk.

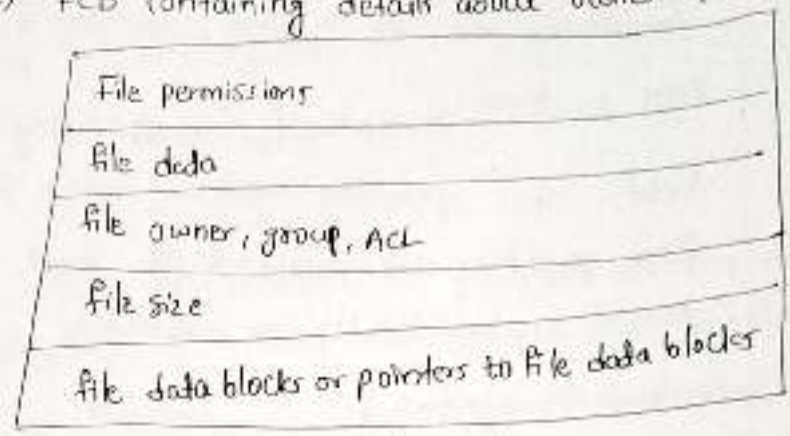
→ A boot control block can contain information needed by the system to boot an operating system from that volume. if the disk does not contain an operating system, this block can be empty. in UFS, it is called boot block, in NTFS, it is the partition boot sector.

→ A volume control block contains volume details, such as the number of blocks in the partition, the size of the blocks, a free block count and free-block pointers. In UFS, this is called a superblock, in NTFS, it is stored in the master file table.

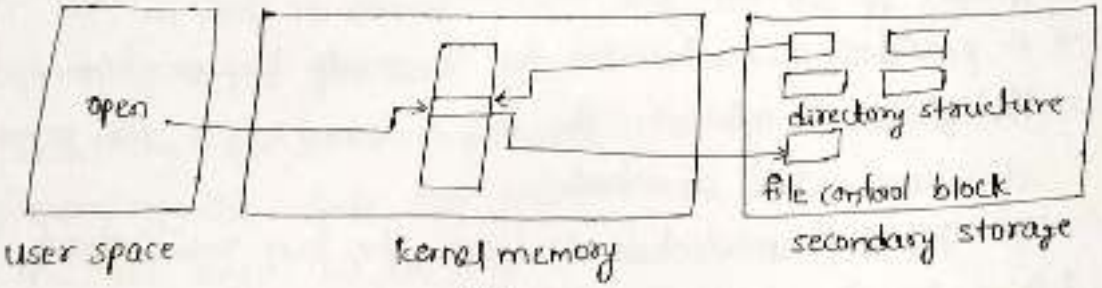
[UFS- Unix file system, NTFS - new Technology file system]

→ A directory structure 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.

File control Block(FCB) FCB containing details about ownership, size, permissions, dates etc.

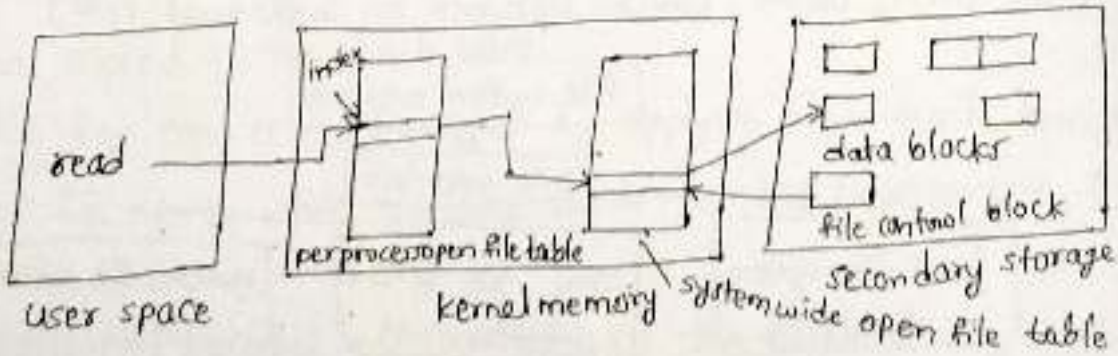


File control block



file open

→ above figure referes to opening a file plus buffer hold data blocks from secondary storage.



file read

→ above figure referes to reading a file as per process open file tables containing a pointer to the system open file table as well as some other information.

Partitioning and mounting:

→ A disk can be divided into multiple partitions, or a partition can span multiple disks. each partition can either is raw, containing no file system, containg a file system

Root partition: It contains the operating system, other partitions can hold other operating systems, other file system, or be raw, Root partition are mounted at boot time. other partitions can mount automatically or manually.

virtual file system (VFS):

→ It separates the file system generic operations from their implementatio by defining a clean VFS interface.

→ It provides a mechanism for uniquely representing a file throughout a network. the VFS is based on a file-representation structure called a vnode.

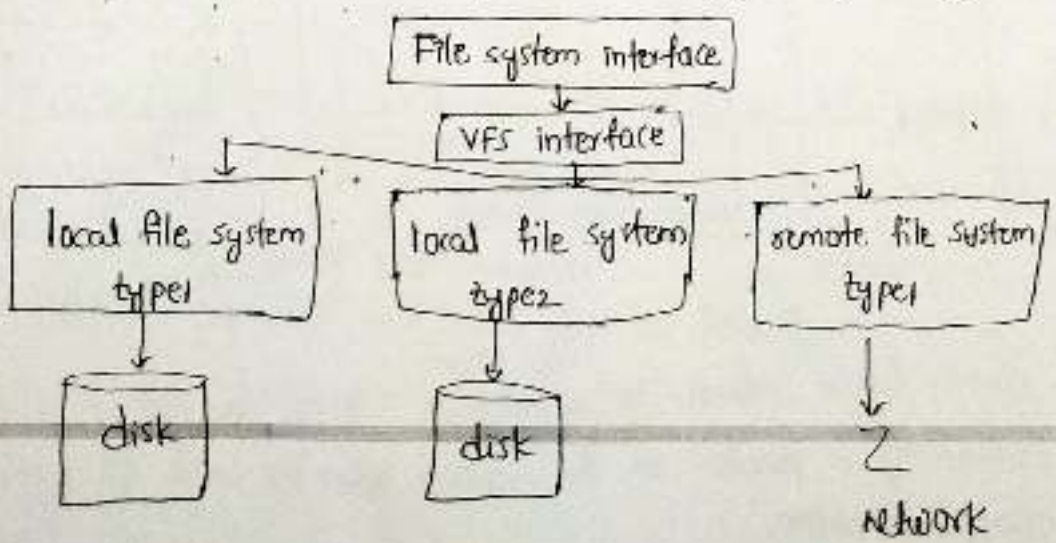
The VFS architecture in Linux the four main object types defined by the linux VFS are

→ The inode object, which represents an individual file

→ The file object, which represents an open file

→ The superblock object, which represents an entire file system.

→ The dentry object, which represents an individual entry.



virtual file system.

Directory implementation:

The Directory implementation algorithms are classified according to the data structure they are using. the selection of an appropriate directory implementation algorithm may significantly affect the performence of the system.

1. Linear List 2. hash table

Linear List:

→ In Lineart List, each file contains the pointers to the data blocks which are assigned to it and the next file in the directory.

→ When a new file is created, then the entire list is checked whether the new file name is matching to a existing file name or not in case, it doesn't exist, the file can be created at the beginning or at the end.

→ Searching for a unique name is a big concern because traversing the whole list takes time.

→ The list needs to be traversed in case of every operation (creation, del, updating etc) on the files, so the file system becomes inefficient.

Hash table:

→ A key-value pair for each file in the directory gets generated and stored in the hash table.

→ The key can be determined by applying the hash function on the file name while the key points to the corresponding file stored in the directory.

→ searching becomes efficient due to the fact i.e, only hash table entries are checked using the key.

Allocation methods / File Allocation methods:

The allocation methods define how the files are stored in the disk blocks. there are 3 main disk space or allocation methods.

- Contiguous Allocation
- Linked Allocation
- Indexed Allocation

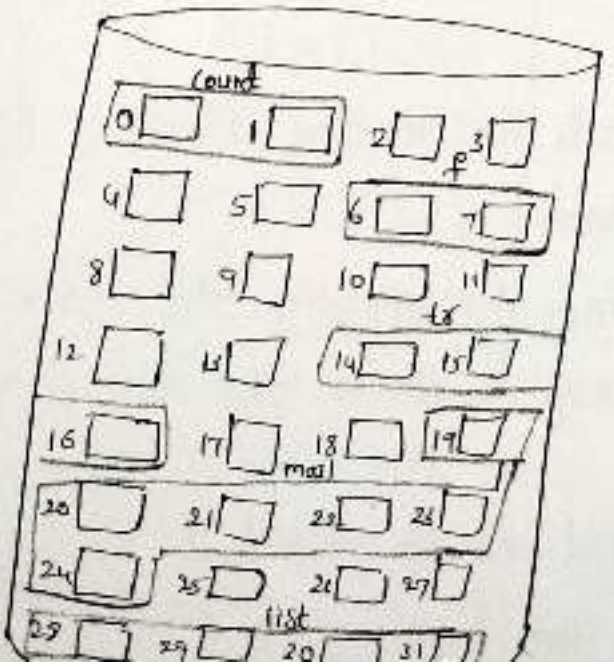
The main aim of file allocation methods is to provide efficient disk space utilization, fast access to the file blocks.

Contiguous Allocation:

→ Each file occupies a contiguous set of blocks on the disk eg: if a file requires 'n' blocks and starting location 'b', then it occupies blocks b, b+1, b+2 ... b+n-1.

→ so, it required address of starting block and length of the allocated portion.

→ starting block - 19, length - 6 so it occupies 19, 20, 21, 22, 23, 24 blocks.



Directory

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

Advantage:

→ There is flexible to increase the file size.

→ It doesn't suffer from external fragmentation, better utilization in terms of memory.

Disadvantage:

→ To access the each block it need time, it makes linked allocation slower

→ it doesn't support random or direct access

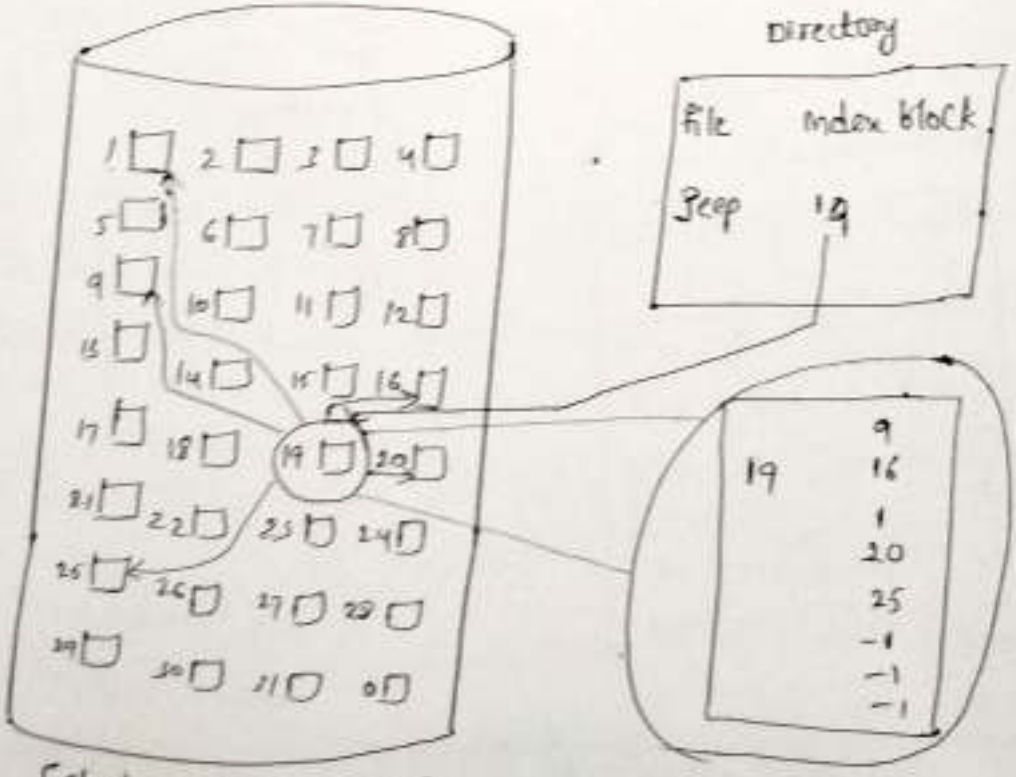
→ pointers required in LLA

Indexed Allocation:

→ It contains a special block known as index block. and index block contains the pointer to all the blocks occupied by a file

→ Each file has its own index block

→ the ith entry in the index block contains the disk address of of the ith file block.



Indexed Allocation of disk space

Advantages:

→ Both Direct and sequential allocation supported by this. for direct access, the address of kth block of the file which starts at block b can easily obtained as (b+k).

→ It extremely fast.

Disadvantages:

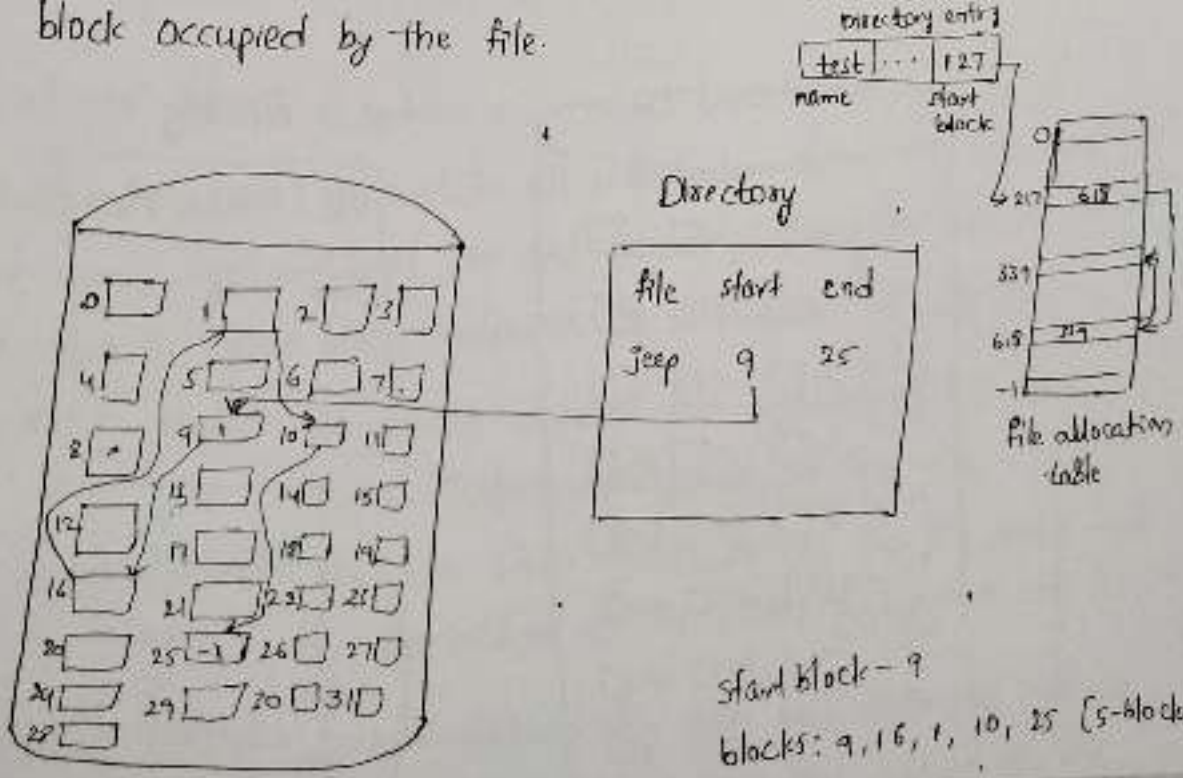
→ It suffers from both internal and external fragmentation. this makes it inefficient in terms of memory utilization.

→ Increasing file size is difficult because it depends on the availability of contiguous memory at a particular instance.

Linked List Allocation:

→ In LLA, each file is a linked list of disk blocks which need not be contiguous.

→ The disk entry contains a pointer to the starting and the ending file block. each block contains a pointer to the next block occupied by the file.



Linked Allocation of diskspace

Start block - 9

blocks: 9, 16, 1, 10, 25 (5-blocks)

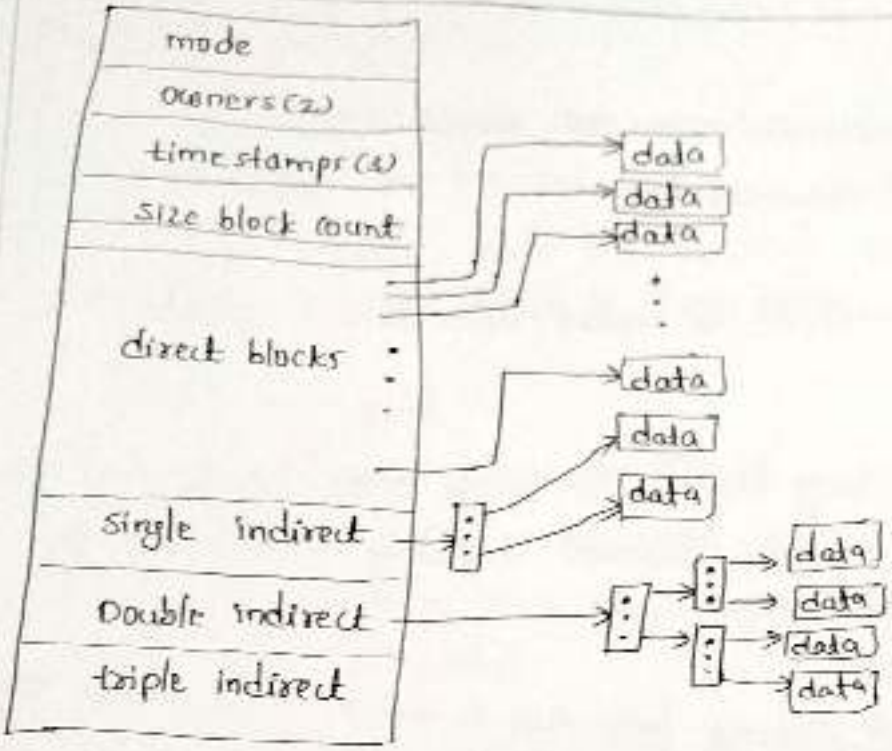
starting block: 1

ending block: -1

Advantages:
→ It support direct access of blocks
→ no external fragmentation
Disadvantages:
→ The pointer overhead for indexed allocation is greater than linked allocation.

For very large files, single index block may not be able to hold all the pointers. following mechanisms can be used to resolve this

1. Linked scheme:
This scheme links two or more index blocks together for holding the pointers. Every index block would then contain a pointer or the address to the next index block
2. Multilevel index: First level index block is used to point to the second level index blocks which points to the disk blocks occupied by the file.
3. Combined Schema: it contain a special block called the inode (Information node), it include all information about the file such a the name, size, authority, etc and remaing space of inode is used to store the Disk block addresses which contains actual file
- direct blocks: the pointer contains the address of the disk blocks that contains data of the file.
- Single indirect: The disk block does not contain the file data bu the disk address of the blocks that contain the file data.
- Double indirect: do not contain file data but the disk address of the blocks that contain the address of the blocks containtaing the file data.



Inode

Free space Management:

Most of the systems disk space is limited, we need to reduce the space from deleted files for new files. To keep track of free disk space, the system maintains a free-space list. the free space list records all free disk blocks - those are not allocated to some file or directory. if we create a new file it allocate space to new file. and that space is removed from the free space list.

The free space list can be implemented mainly as

- Bitmap or Bit vector
- linked list
- Grouping
- Counting

Bitmap or BitVector:

It is a collection or series of bits where each bit corresponds to a disk block the bit can take two values 0 and 1.

0 - indicates that the block is allocated

1 - indicates block is free block

it can be represented as 16 bitmap 0000111000000110

Advantages:

- simple to understand
- finding the 1st free block is efficient. it requires scanning the words in a bitmap for a non-zero word (zero valued word has all bits '0') The 1st free block is then found by scanning for the first 1 bit in the non-zero word.

The block number can be calculated as

(number of bits per word) * (number of 0-value words) + offset of first 1 bit.

In above diagram we get 16 bit map 0000111 0000 0110

1-word = 8 bits

2-word

The block number = 8 X 0 + 5 — 1st word — non zero bit is 5

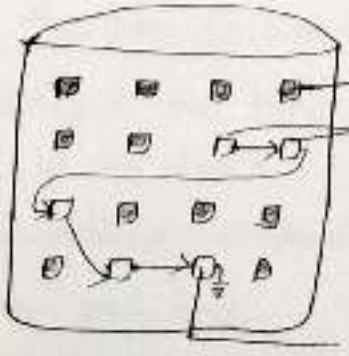
No of bits in word — all bits are not zero in above words

Linked List:

In bitmap if we increase the disk size, it is not easy to maintain free disks blocks

In Linked List free disks blocks contains a pointer to the next free block. The block number of every 1st disk is stored at separate location on disk.

drawback of this method is the I/o required for free space list traversal.



Grouping: it stores the address of free blocks in the first free block If n free blocks, out of these n blocks, the first n-1 blocks are actually free and last block contains the address of the next free n blocks. advantage is free disk blocks can be found easily.

counting:

This approach stores the address of the first free disk block and a number 'n' of free contiguous disk blocks that follow the first block. Every entry in list contain address of 1st free block and number 'n'.

Efficiency and performance:

efficiency dependent on disk allocation and directory algorithms types of data kept in file's directory entry.

Performance:

→ disk cache - separe section of main memory for frequently used blocks.

→ free behind and read-ahead - techniques to optimize sequential access.

→ improve pc performence by dedicating section of memory as vir disk, or RAM disk.

Mass storage structure :

Secondary storage devices are non-volatile. means the stored data will be intact even if the system is turned off.

→ secondary storage is also called auxilary storage.
→ secondary storage is less expensive when compared to primary memory like RAMs.
→ The speed of the secondary storage is also lesser than that of primary storage.
→ The data which is less frequently accessed is kept in the secondary storage.

Examples are magnetic disks, magnetic tapes and removable drives

Magnetic Disk:

→ A magnetic disk contains several platters. each platter is divided into circular shaped tracks.
→ The length of the tracks near the centre is less than the length of the tracks from the centre. Each track is further divided into sectors.
→ Tracks of the same distance from centre from a cylinder A read-write head is used to read data from a sector of the magnetic disk.
→ The speed can be measured as transfer rate and random access time.
→ Transfer rate: The rate at which the data moves from disk to the computer. the sum of the seek time and rotational latency is called random access time
→ seek time: time taken by the arm to move to the required track. rock: rotational latency is the time taken by the arm to reach the required sector in the track.
→ data is logically arranged and addressed as an array of block of fixed size The size of a block is 512 or 1024 bytes.