# UNIT V

**FILE SYSTEM INTERFACE**

# The Concept Of a File

A file is a named collection of related information that is recorded on secondary storage. The information in a file is defined its creator. Many different types of information may be stored in a file.

### File attributes:-

A file is named and for the user’s convince is referred to by its name. A name is

usually a string of characters. One user might create file, where as another user might edit that file by specifying its name. There are different types of attributes.

1. **name:-** the name can be in the human readable form.
2. **type:-** this information is needed for those systems that support different types. 3)**location:-** this information is used to a device and to the location of the file on that device.

4)**size:-** this indicates the size of the file in bytes or words. 5)**protection:-**

### 6)time,date, and user identifications:-

the information about all files is kept in the directory structure, that also resides on secondary storage.

### File operations:- Creating a file:-

Two steps are necessary to create a file first, space in the file system must be found for the file. Second , an entry for the new file must be made in the directory. The directory entry records the name of the file and the location in the system.

### Writing a file:-

To write a file give the name of the file, the system search the directory to find the location of the file. The system must keep the *writer* 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.

**Reading a file:-** to read from a file, specifies the name of the file and directory is search for the associated directory entry, and the system needs to keep *read* pointer to the location in the file where the next read is to take place. Once the read has taken place, read pointer is updated.

### Repositioning with in a file:-

The directory is searched for the appropriate entry and the current file position is set to given value. this is also known as a file seek.

**Deleting a file:-** to delete a file , we search the directory for the name file. Found that file in the directory entry, we release all file space and erase the directory entry.

**Truncate a file:-** this function allows all attributes to remain unchanged(except for file length) but for the file to be reset to length zero.

**Appending:-** add new information to the end of an existing file .

**Renaming:-** give new name to an existing file.

**Open a file:-**if file need to be used, the first step is to open the file, using the *open*

system call.

**Close:-** close is a system call used to terminate the use of an already used file.

### File Types:-

A common technique for implementing file type is to include the type as part of the file name. The name is split in to two parts

1. the name 2) and an extension .

the system uses the extension to indicate the type of the file and the type of operations that can be done on that file.

### : ACCESSMETHODS:-

**There are** several ways that the information in the file can be accessed. **1)sequential method 2) direct access method 3) other access methods.** 1)sequential access method:-

the simplest access method is S.A. information in the file is processed in order, one after the other. the bulk of the operations on a file are reads & writes. It is based on a tape model of a file. Fig 10.3

1. Direct access:- or relative access:-

a file is made up of fixed length records, that allow programs to read and write record rapidly in no particular order. For direct access, file is viewed as a numbered sequence of blocks or records. A direct access file allows, blocks to be read & write. So we may read block15, block 54 or write block10. there is no restrictions on the order of reading or writing for a direct access file. It is great useful for immediate access to large amount of information.

The file operations must be modified to include the block number as a parameter. We have read n, where n is the block number.

1. other access methods:-

the other access methods are based on the index for the file. The indexed contain pointers to the various blocks. To find an entry in the file , we first search the index and then use the pointer to access the file directly and to find the desired entry. With large files. The index file itself, may become too large to be kept in memory. One solution is to create an index for the index file. The primary index file would contain pointers to secondary index files which would point to the actual data iteams

### Directory Structures:-

operations that are be on a directory (read in text book)

### single level directory:-

the simple directory structure is the single level directory. All files are contained in

the same directory. Which is easy to understand. Since all files are in same directory, they must have unique names.

In a single level directory there is some limitations. When the no.of files

increases or when there is more than one user some problems can occurs. If the no.of files increases, it becomes difficult to remember the names of all the files. FIG 10.7 **Two-level directory:-**

The major disadvantages to a single level directory is the confusion of file names between different users. The standard solution is to create separate directory for each user.

In 2-level directory structure, each user has her own user file directory(ufd). Each ufd has a similar structure, the user first search the master file directory . the mfd is indexed by user name and each entry point to the ufd for that user.fig 10.8

To create a file for a user, the O.S search only that user’s ufd to find whether another file of that name exists. To delete a file the O.S only search to the local ufd and it can not accidentally delete another user’s file that has the same name.

This solves the name collision problem, but it still have another. This is disadvantages when the user wants to cooperate on some task and to access one another’s file . some systems simply do not allow local user files to be accessed by other user.

Any file is accessed by using path name. Here the user name and a file name defines a path name.

Ex:- user1/ob

In MS-DOS a file specification is C:/directory name/file name **Tree structured directory:-**

This allows users to create their own subdirectories and to organize their files accordingly. here the tree hasa root directory. And every file in the system has a unique path name. A path name is the path from the root, through all the subdirectories to a specified file.FIG 10.9.

A directory contains a set of subdirectories or files. A directory is simply another file, but it is treated in a special way. Here the path names can be of two types. 1)absolute path and 2) relative path.

An absolute path name begins at the root and follows a path down to the specified file, giving the directory name on the path.

Ex:- root/spell/mail/prt/first.

A relative pathname defines a path from the current directory ex:- prt/first is relative path name.

### A cyclic- graph directory:-

Consider two programmers who are working on a joint project. The files associated with that project can be stored in a sub directory , separating them from

other projects and files of the two programmers. The common subdirectory is shared by both programmers. A shared directory or file will exist in the file system in two places at once. Notice 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 others copy with a shared file there is only one actual file, so any changes made by one person would be immediately visible to the other.

A tree structure prohibits the sharing of files or directories. An acyclic graph allows directories to have shared subdirectories and files

FIG 10.10 . it is more complex and more flexiable. Also several problems may occurs at the traverse and deleting the file contents.

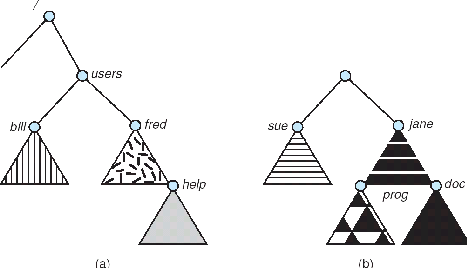
.

# :File System Mounting

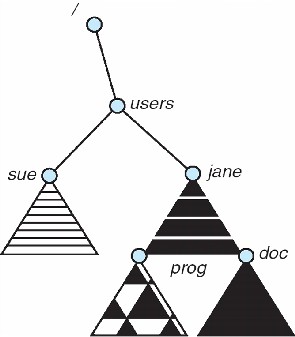
A file system must be **mounted** before it can be accessed

A unmounted file system (i.e. Fig. 11-11(b)) is mounted at a **mount point**

## Existing. (b) Unmounted Partition

* + 1. 

**Mount Point**



# :File Sharing

Sharing of files on multi-user systems is desirablenSharing may be done through a **protection** schemenOn distributed systems, files may be shared

across a networknNetwork File System (NFS) is a common distributed filesharing method

## File Sharing – Multiple Users

**User IDs** identify users, allowing permissions and protections to be perusern**Group IDs** allow users to be in groups, permitting group access rights

## File Sharing – Remote File Systems

nUses networking to allow file system access between systems

lManually via programs like FTP

lAutomatically, seamlessly using **distributed file systems**

lSemi automatically via the **world wide web**

n**Client-server** model allows clients to mount remote file systems from servers

lServer can serve multiple clients

lClient and user-on-client identification is insecure or complicated

l**NFS** is standard UNIX client-server file sharing protocol

l**CIFS** is standard Windows protocol

lStandard operating system file calls are translated into remote calls nDistributed Information Systems **(distributed naming services)** such as LDAP, DNS, NIS, Active Directory implement unified access to information needed for remote computing

## File Sharing – Failure Modes

Remote file systems add new failure modes, due to network failure, server failure

Recovery from failure can involve state information about status of each remote request

Stateless protocols such as NFS include all information in each request, allowing easy recovery but less security

**File Sharing – Consistency Semantics**

n**Consistency semantics** specify how multiple users are to access a shared file simultaneously

lSimilar to Ch 7 process synchronization algorithms

Tend to be less complex due to disk I/O and network latency (for remote file systems

lAndrew File System (AFS) implemented complex remote file sharing semantics

lUnix file system (UFS) implements:

Writes to an open file visible immediately to other users of the same open file

Sharing file pointer to allow multiple users to read and write concurrently

lAFS has session semantics

Writes only visible to sessions starting after the file is closed

# :Protection

File owner/creator should be able to control:

lwhat can be done

lby whomnTypes of access

### lRead lWrite lExecute lAppend lDelete lList

**Protection:-**]

When the information is kept in the system the major worry is its protection from the both physical damage (Reliability) and improper access(Protection).

The reliability is generally provided by duplicate copies of files.

The protection can be provided in many ways . for some single system user, we might provide protection by physically removing the floppy disks . in a multi-user systems, other mechanism are needed.

### 1) types of access:-

if the system do not permit access to the files of other users, protection is not needed. Protection mechanism provided by controlling accessing. This can be provided by types of file access. Access is permitted or denied depending on several factors. Suppose we mentioned read that file allows only for read .

Read:- read from the file. Write:- write or rewrite the file.

Execute:- load the file in to 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.

### FILE SYSTEM IMPLEMENTATION

* 1. **:File allocation methods:-**

There are 3 major methods of allocating disk space.

### 1) Contiguous allocation:-

1. The contiguous allocation method requires each file to occupy a set of contiguous block on the disk.
2. Contiguous allocation of a file is defined by the disk address and length of the first block. If the file is ‘n’ block long and starts at location ‘b’ , then it occupies blocks b,b+1,b+2,…..,b+n-1;
3. The directory entry for each file indicates the address of the starting block and length of the area allocated for this file. Fig **11.3**
4. Contiguous allocation of file is very easy to access. For the *sequential access* , the file system remembers the disk address of the last block referenced and, when necessary read next block. For *direct access* to block ‘i’ of a file that starts at block ‘b’ , we can immediately access block b+i. Thus both sequential and direct access can be supported by contagious allocation.
5. One difficulty with this method is finding space for a new file.
6. Also there are many problems with this method
   1. **external fragmentation:-** files are allocated and deleted , the free disk space is broken in to little pieces. The E.F exists when free space is broken in to chunks(large piece) and these chunks are not sufficient for a request of new file.

There is a solution for E.F i.e compaction. All free space compact in to one contiguous space. But the cost of compaction is time.

* 1. Another problem is determining how much space is needed for a file. When file is created the creator must specifies the size of

that file. This becomes to big problem. Suppose if we allocate too little space to a file , some times it may not sufficient.

Suppose if we allocate large space some times space is wasted.

* 1. Another problem is if one large file is deleted, that large space is becomes to empty. Another file is loaded in to that space whose size is very small then some space is wasted . that wastage of space is called internal fragmentation.

### 2) Linked allocation:-

1. Linked allocation solves all the problems of contagious allocation. With linked allocation , each file is a linked list of disk blocks, the disk block may be scattered any where on the disk.
2. The directory contains a pointer to the first and last blocks of the file. **Fig11.4** Ex:- a file have five blocks start at block 9, continue at block 16,then block 1, block 10 and finally block 25. each block contains a ponter to the next block. These pointers are not available to the user.
3. To create a new file we simply creates a new entry in directory. With linked allocation, each directory entry has a pointer to the first disk block of the file.
4. There is no external fragmentation with linked allocation. Also there is no need to declare the size of a file when that file is created. A file can continue to grows as long as there are free blocks.
5. But it have disadvantage. The major problem is that it can be used only for sequential access-files.
6. To find the I th block of a file , we must start at the beginning of that file, and follow the pointers until we get to the I th block. It can not support the direct access.
7. Another disadvantage is it requires space for the pointers. If a pointer requires 4 bytes out of 512 byte block, then 0.78% of disk is being used for pointers, rather than for information.
8. The solution to this problem is to allocate blocks in to multiples, called clusters and to allocate the clusters rather than blocks.
9. Another problem is reliability. The files are linked together by pointers scattered all over the disk what happen if a pointer were lost or damaged. **FAT( file allocation table):-**

An important variation on the linked allocation method is the use of a file allocation table.

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 entry contains the block number of the first block of the file. The table entry contains the block number then contains the block number of the next block in the file. This chain continuous until the last block, which has a special end of file values as the table entry. Unused blocks are indicated by a ‘0’ table value. Allocation a new block to a file is a simple. First finding the first 0-value table entry, and replacing the previously end of file value with the address of the new block. The 0 is then replaced with end of file value.

### Fig 11.5

**3)Indexed allocation:-**

1. linked allocation solves the external fragmentation and size declaration problems of contagious allocation. How ever in the absence of a FAT , linked allocation can not support efficient direct access.
2. The pointers to the blocks are scattered with the blocks themselves all over the disk and need to be retrieved in order.
3. Indexed allocation solves this problem by bringing all the pointers together in to one location i.e *the index block.*
4. 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 ith block of the file.
5. The directory contains the address of the index block. **Fig 11.6**

To read the ith block we use the pointer in the ith index block entry to find and read the desired block.

1. When the file is created, all pointers in the index block are set to nil. When the ith block is first written, a block is obtained from the free space manager, and

its address is put in the ith index block entry.

1. It supports the direct access with out suffering from external fragmentation, but it suffer from the wasted space. The pointer overhead of the index block is generally greater than the pointer over head of linked allocation.

### :Free space management:-

1. to keep track of free disk space, the system maintains a free space list. The free space list records all disk blocks that are free.
2. To create a file we search the free space list for the required amount of space, and allocate that space to the new file. This space is then removed from the free space list.
3. When the file is deleted , its disk space is added to the free space list. There are many methods to find the free space.

### bit vector:-

The free space list is implemented as a bit map or bit vector. Each block is represented by 1 bit. If the block is free the bit is 1 if the block is allocated the bit is 0.

**Ex:-** consider a disk where blocks 2,3,4,5,8,9,10,11,12,13,17,18,25, are free and rest of blocks are allocated the free space bit map would be **001111001111110001100000010000…**…..

the main advantage of this approach is that it is relatively simple and efficient to find the first free block or ‘n’ consecutive free blocks on the disk

### Linked list:-

Another approach is to link together all the free disk blocks, keeping a pointer

to the first free block in a special location on the disk and caching it in memory. This first block contain a pointer to the next free disk block, and so on.

How ever this scheme is not efficient to traverse the list, we must read each block, which requires I/O time.

Disk space is also wasted to maintain the pointer to next free space.

### Grouping:-

Another method is 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. **Fig 11.8**

Advantages:- the main advantage of this approach is that the addresses of a large no.of blocks can be found quickly.

### Counting:-

Another approach is counting. Generally several contiguous blocks may be allocated or freed simultaneously. Particularly when space is allocated with the contiguous allocation algorithm rather than keeping a list of ‘n’ free disk address. We can keep the address of 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.

### :Directory Implementation:-

**1) Linear list:-**

1. the simple method of implement ting a directory is to use a linear list of file names with pointers to the data blocks.
2. A linear list of directory entries requires a linear search to find a particular entry.
3. This method is simple to program but is time consuming to execute.
4. To create a new file, we must first search the directory to be sure that no existing file has the same name. Then, we add a new entry at the end of the directory.
5. To delete a file we search the directory for the named file, then release the space allocated to it.
6. To reuse directory entry, we can do one of several things.
7. We can mark the entry as unused or we can attach it to a list of free directory entries.

Disadvantage:- the disadvantage of a linear list of directory entries is the linear search to find a file.

**MASS-STORAGE STRUCTURE**

# Mass-Storage Systems

nDescribe the physical structure of secondary and tertiary storage devices and the resulting effects on the uses of the devicesnExplain the performance characteristics of mass-storage devicesnDiscuss operating-system services provided for mass storage, including RAID and HSM

# :Overview of Mass Storage Structure

Magnetic disks provide bulk of secondary storage of modern computers Drives rotate at 60 to 200 times per second

Transfer rate is rate at which data flow between drive and computer

Positioning time (random-access time) is time to move disk arm to desired cylinder (seek time) and time for desired sector to rotate under the disk head (rotational latency) Head crash results from disk head making contact with the disk surface

That’s bad

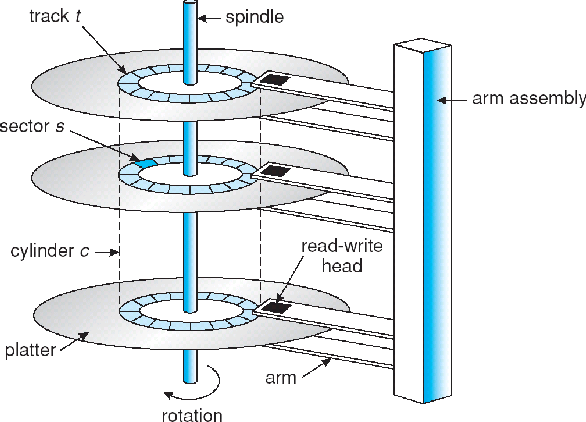
Disks can be removable

Drive attached to computer via I/O bus

Busses vary, including EIDE, ATA, SATA, USB, Fibre Channel, SCSI

Host controller in computer uses bus to talk to disk controller built into drive or storage array

## Moving-head Disk Mechanism



Magnetic tape

Was early secondary-storage medium

Relatively permanent and holds large quantities of data Access time slow

Random access ~1000 times slower than disk

Mainly used for backup, storage of infrequently-used data, transfer medium between systems

Kept in spool and wound or rewound past read-write head Once data under head, transfer rates comparable to disk 20-200GB typical storage

Common technologies are 4mm, 8mm, 19mm, LTO-2 and SDLT

# :Disk Structure

Disk drives are addressed as large 1-dimensional arrays of logical

blocks, where the logical block is the smallest unit of transfernThe 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially

Sector 0 is the first sector of the first track on the outermost cylinder

Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost **8.3:Disk Attachment**

Host-attached storage accessed through I/O ports talking to I/O busses SCSI itself is a bus, up to 16 devices on one cable, SCSI initiator requests operation and SCSI targets perform tasks

Each target can have up to 8 logical units (disks attached to device controller FC is high-speed serial architecture

Can be switched fabric with 24-bit address space – the basis of storage area networks (SANs) in which many hosts attach to many storage units

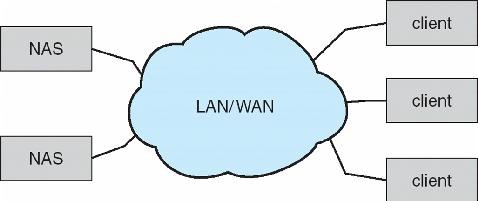
Can be arbitrated loop (FC-AL) of 126 devices

## Network-Attached Storage

Network-attached storage (NAS) is storage made available over a network rather than over a local connection (such as a bus)

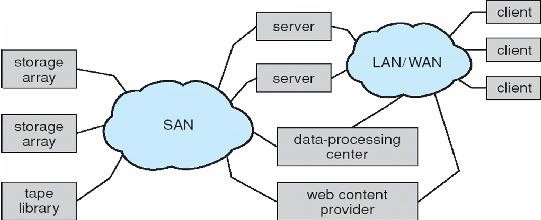
NFS and CIFS are common protocols

Implemented via remote procedure calls (RPCs) between host and storage New iSCSI protocol uses IP network to carry the SCSI protocol



## Storage Area Network

Common in large storage environments (and becoming more common) Multiple hosts attached to multiple storage arrays – flexible



# :Disk Scheduling

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

Access time has two major components

Seek time is the time for the disk are to move the heads to the cylinder containing the desired sector

Rotational latency is the additional time waiting for the disk to rotate the desired sector to the disk head

Minimize seek time

Seek time » seek distance

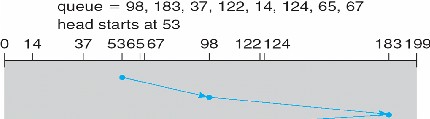
Disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer Several algorithms exist to schedule the servicing of disk I/O requests nWe illustrate them with a request queue (0-199)

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

Head pointer 53

## FCFS

Illustration shows total head movement of 640 cylinders



## SSTF

Selects the request with the minimum seek time from the current head position SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests

nIllustration shows total head movement of 236 cylinders

## SCAN

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.nSCAN algorithm Sometimes called the elevator algorithm

Illustration shows total head movement of 208 cylinders

## C-SCAN

Provides a more uniform wait time than SCAN

The head moves 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

Treats the cylinders as a circular list that wraps around from the last cylinder to the first one

## C-LOOK

Version of C-SCAN

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 **Selecting a Disk-Scheduling Algorithm**

SSTF is common and has a natural appeal

SCAN and C-SCAN perform better for systems that place a heavy load on the disk

Performance depends on the number and types of requests

Requests for disk service can be influenced by the file-allocation method The disk-scheduling algorithm should be written as a separate module of the

operating system, allowing it to be replaced with a different algorithm if necessary Either SSTF or LOOK is a reasonable choice for the default algorithm

## Disk Management

Low-level formatting, or physical formatting — Dividing a disk into sectors that the disk controller can read and write

To use a disk to hold files, the operating system still needs to record its own data structures on the disk

Partition the disk into one or more groups of cylinders Logical formatting or “making a file system”

To increase efficiency most file systems group blocks into clusters

Disk I/O done in blocks

File I/O done in clusters Boot block initializes system

The bootstrap is stored in ROM Bootstrap loader program

Methods such as sector sparing used to handle bad blocks

# Booting from a Disk in Windows 2000

* 1. **:Swap-Space Management**

Swap-space — Virtual memory uses disk space as an extension of main memory

Swap-space can be carved out of the normal file system, or, more commonly, it can be in a separate disk partition

Swap-space management

4.3BSD allocates swap space when process starts; holds text segment (the program) and data segment

Kernel uses swap maps to track swap-space use

Solaris 2 allocates swap space only when a page is forced out of physical

memory, not when the virtual memory page is first created

## Data Structures for Swapping on Linux Systems

**RAID Structure**

RAID – multiple disk drives provides reliability via redundancynIncreases the mean time to failurenFrequently combined with NVRAM to improve write performance

RAID is arranged into six different levels

Several improvements in disk-use techniques involve the use of multiple disks working cooperativelynDisk striping uses a group of disks as one storage unitnRAID schemes improve performance and improve the reliability of the storage system by storing redundant data

Mirroring or shadowing (RAID 1) keeps duplicate of each disk

Striped mirrors (RAID 1+0) or mirrored stripes (RAID 0+1) provides high

performance and high reliabilitylBlock interleaved parity (RAID 4, 5, 6) uses much less redundancy RAID within a storage array can still fail if the array fails, so automatic

replication of the data between arrays is common

Frequently, a small number of hot-spare disks are left unallocated, automatically replacing a failed disk and having data rebuilt onto them

## RAID (0 + 1) and (1 + 0)

**Extensions**

RAID alone does not prevent or detect data corruption or other errors, just disk failures

Solaris ZFS adds checksums of all data and metadata

Checksums kept with pointer to object, to detect if object is the right one and whether it changed

Can detect and correct data and metadata corruption ZFS also removes volumes, partititions

Disks allocated in pools

Filesystems with a pool share that pool, use and release space like “malloc” and “free” memory allocate / release calls

## ZFS Checksums All Metadata and Data Traditional and Pooled Storage

**Stable-Storage Implementation**

Write-ahead log scheme requires stable storagenTo implement stable storage: Replicate information on more than one nonvolatile storage media with independent failure modes

Update information in a controlled manner to ensure that we can recover the stable data after any failure during data transfer or recovery

## Tertiary Storage Devices

Low cost is the defining characteristic of tertiary storagenGenerally, tertiary storage is built using removable medianCommon examples of removable media are floppy disks and CD-ROMs; other types are available

## Removable Disks

Floppy disk — thin flexible disk coated with magnetic material, enclosed in a protective plastic caselMost floppies hold about 1 MB; similar technology is used for removable disks that hold more than 1 GB

Removable magnetic disks can be nearly as fast as hard disks, but they are at a greater risk of damage from exposure

A magneto-optic disk records data on a rigid platter coated with magnetic material

Laser heat is used to amplify a large, weak magnetic field to record a bit Laser light is also used to read data (Kerr effect)

The magneto-optic head flies much farther from the disk surface than a magnetic disk head, and the magnetic material is covered with a protective layer of plastic or glass; resistant to head crashesnOptical disks do not use magnetism; they employ special materials that are altered by laser light

## WORM Disks

The data on read-write disks can be modified over and over

WORM (“Write Once, Read Many Times”) disks can be written only once Thin aluminum film sandwiched between two glass or plastic platters

To write a bit, the drive uses a laser light to burn a small hole through the aluminum; information can be destroyed by not altered

Very durable and reliable

Read-only disks, such ad CD-ROM and DVD, com from the factory with the data pre-recorded

## Tapes

Compared to a disk, a tape is less expensive and holds more data, but random access is much slower

Tape is an economical medium for purposes that do not require fast random access, e.g., backup copies of disk data, holding huge volumes of data Large tape installations typically use robotic tape changers that move tapes between tape drives and storage slots in a tape library

stacker – library that holds a few tapes silo – library that holds thousands of tapes

A disk-resident file can be archived to tape for low cost storage; the computer can stage it back into disk storage for active use

## Operating System Support

nMajor OS jobs are to manage physical devices and to present a virtual machine abstraction to applicationsnFor hard disks, the OS provides two abstraction:

Raw device – an array of data blockslFile system – the OS queues and schedules the interleaved requests from

several applications

## Application Interface

Most OSs handle removable disks almost exactly like fixed disks — a new cartridge is formatted and an empty file system is generated on the disk Tapes are presented as a raw storage medium, i.e., and application does not not open a file on the tape, it opens the whole tape drive as a raw device Usually the tape drive is reserved for the exclusive use of that application

Since the OS does not provide file system services, the application must decide how to use the array of blocks

Since every application makes up its own rules for how to organize a tape, a tape full of data can generally only be used by the program that created it **Tape Drives**

The basic operations for a tape drive differ from those of a disk drive locate() positions the tape to a specific logical block, not an entire track (corresponds to seek())

The read position() operation returns the logical block number where the

tape head is

The space() operation enables relative motion

Tape drives are “append-only” devices; updating a block in the middle of the tape also effectively erases everything beyond that block

An EOT mark is placed after a block that is written

## File Naming

The issue of naming files on removable media is especially difficult when we want to write data on a removable cartridge on one computer, and then use the cartridge in another computer

Contemporary OSs generally leave the name space problem unsolved for removable media, and depend on applications and users to figure out how to access and interpret the data

Some kinds of removable media (e.g., CDs) are so well standardized that all computers use them the same way

## (Hierarchical Storage Management HSM)

A hierarchical storage system extends the storage hierarchy beyond

primary memory and secondary storage to incorporate tertiary storage — usually implemented as a jukebox of tapes or removable disks

Usually incorporate tertiary storage by extending the file system Small and frequently used files remain on disk

Large, old, inactive files are archived to the jukebox

HSM is usually found in supercomputing centers and other large installations that have enormous volumes of data

## Speed

Two aspects of speed in tertiary storage are bandwidth and latencynBandwidth is measured in bytes per second

Sustained bandwidth – average data rate during a large transfer; # of bytes/transfer time

Data rate when the data stream is actually flowing

Effective bandwidth – average over the entire I/O time, including seek() or

locate(), and cartridge switching Drive’s overall data rate

Access latency – amount of time needed to locate data

Access time for a disk – move the arm to the selected cylinder and wait for the rotational latency; < 35 milliseconds

Access on tape requires winding the tape reels until the selected block reaches the tape head; tens or hundreds of seconds

Generally say that random access within a tape cartridge is about a thousand times slower than random access on disk

The low cost of tertiary storage is a result of having many cheap cartridges share a few expensive drives

A removable library is best devoted to the storage of infrequently used data, because the library can only satisfy a relatively small number of I/O requests per hour

## Reliability

A fixed disk drive is likely to be more reliable than a removable disk or tape drivenAn optical cartridge is likely to be more reliable than a magnetic disk or tapenA head crash in a fixed hard disk generally destroys the data, whereas the failure of a tape drive or optical disk drive often leaves the data cartridge

unharmed

## Cost

Main memory is much more expensive than disk storagenThe cost per megabyte of hard disk storage is competitive with magnetic tape if only one tape is used per drivenThe cheapest tape drives and the cheapest disk drives have had about the same storage capacity over the yearsnTertiary storage gives a cost savings only when the number of cartridges is considerably larger than the number of drives