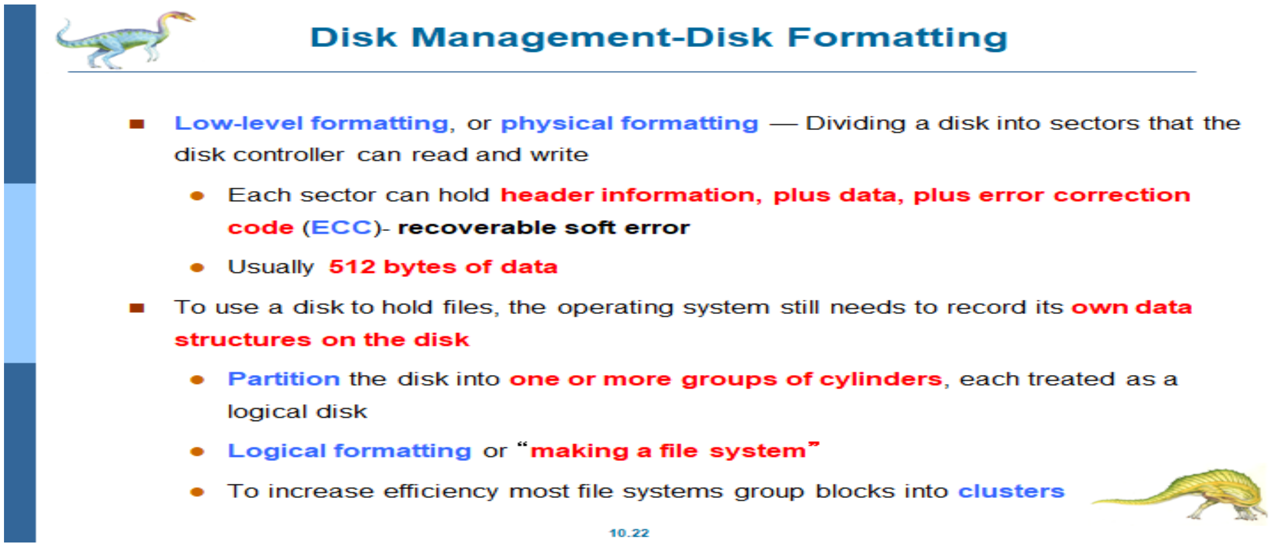
Disk Management

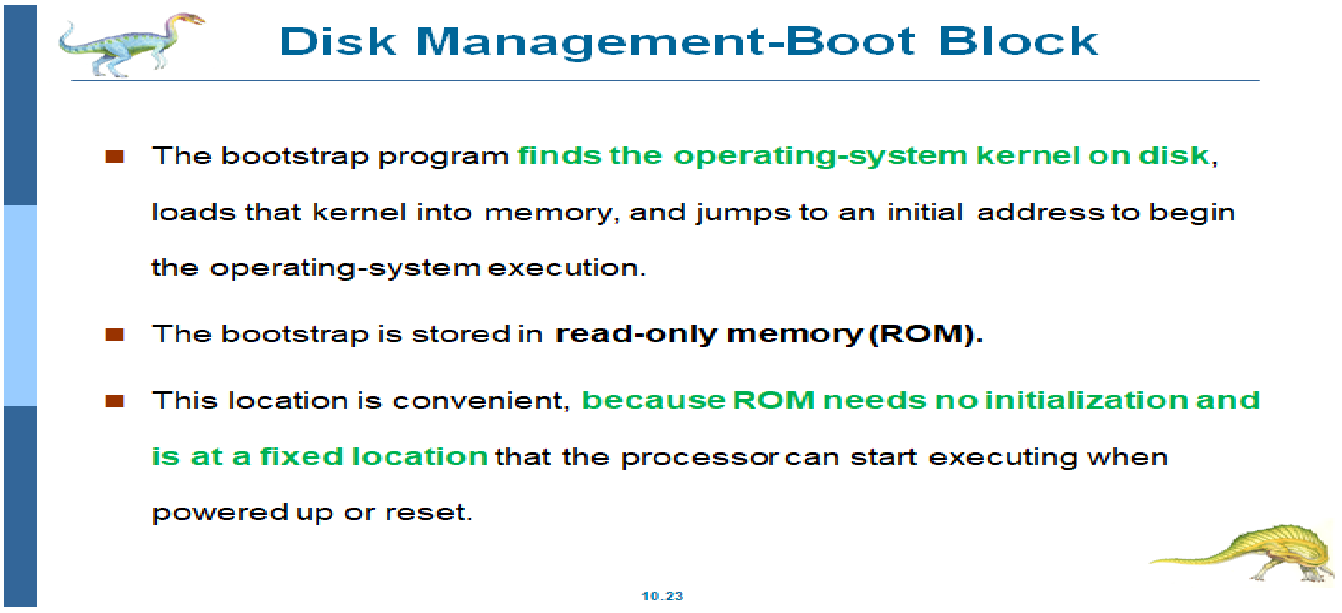
- Disk Formatting
- Boot Block
- Bad Blocks



Disk Management-Disk Formatting

- Low-level formatting, or physical formatting — Dividing a disk into sectors that the disk controller can read and write
  - Each sector can hold header information, plus data, plus error correction code (ECC)- **recoverable soft error**
  - Usually 512 bytes of data
- 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, each treated as a logical disk
  - Logical formatting or "making a file system"
  - To increase efficiency most file systems group blocks into clusters

10.22



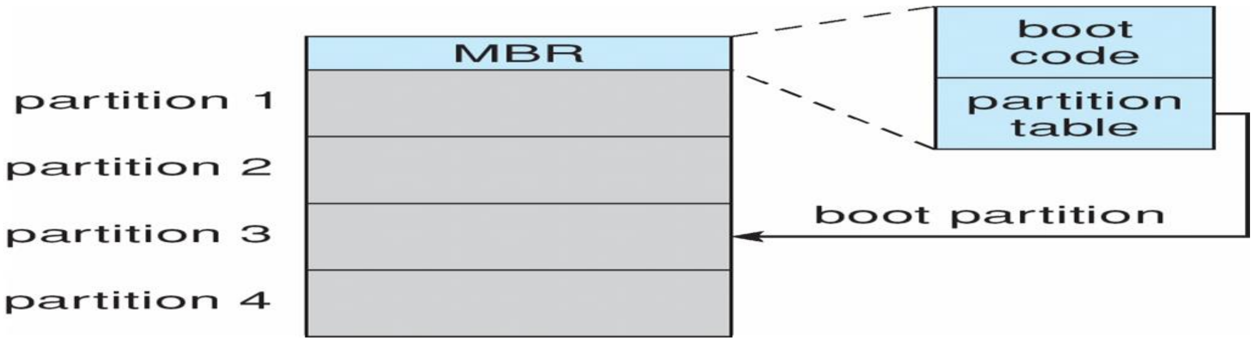
Disk Management-Boot Block

- The bootstrap program finds the operating-system kernel on disk, loads that kernel into memory, and jumps to an initial address to begin the operating-system execution.
- The bootstrap is stored in **read-only memory (ROM).**
- This location is convenient, because ROM needs no initialization and is at a fixed location that the processor can start executing when powered up or reset.

10.23

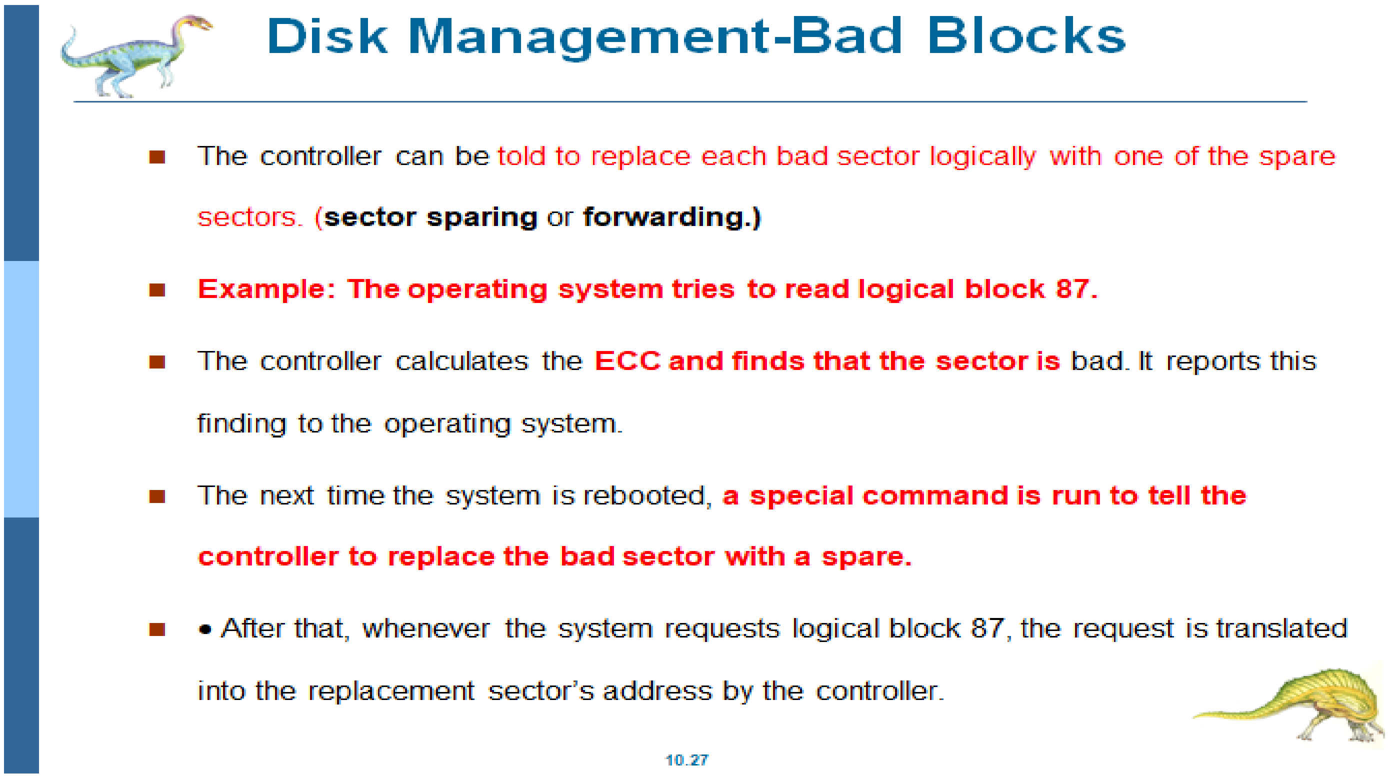
Disk Management-Boot Block

- The problem is that changing this bootstrap code requires changing the ROM hardware chips.
- Most systems store a tiny bootstrap loader program in the boot ROM whose only job is to bring in a full bootstrap program from disk.
- The full bootstrap program is stored in the "boot blocks" at a fixed location on the disk. A disk that has a boot partition is called a boot disk or system disk.



Disk Management-Bad Blocks

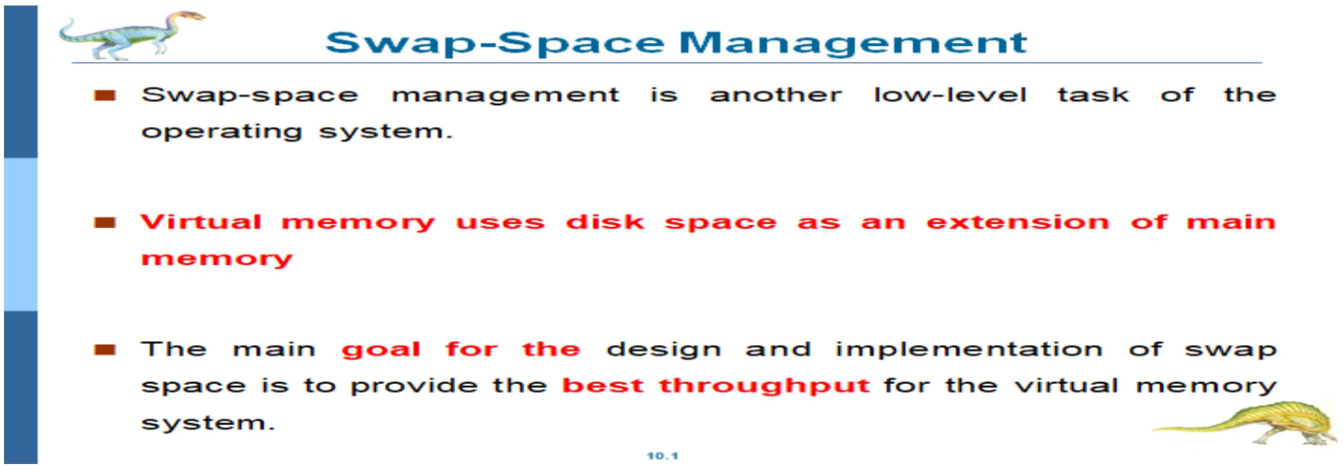
- disks have moving parts and they are prone to failure.
- Most disks even come from the factory with **bad blocks.**
- handled in a variety of ways.
- The controller maintains a list of bad blocks on the disk. The list is initialized during the low-level formatting at the factory
- Low-level formatting also sets aside spare sectors



Disk Management-Bad Blocks

- The controller can be told to replace each bad sector logically with one of the spare sectors. (**sector sparing** or **forwarding.)**
- **Example: The operating system tries to read logical block 87.**
- The controller calculates the **ECC and finds that the sector is** bad. It reports this finding to the operating system.
- The next time the system is rebooted, **a special command is run to tell the controller to replace the bad sector with a spare.**
- • After that, whenever the system requests logical block 87, the request is translated into the replacement sector's address by the controller.

Swap-Space Management



Swap-Space Management

- Swap-space management is another low-level task of the operating system.
- **Virtual memory uses disk space as an extension of main memory**
- The main **goal for the** design and implementation of swap space is to provide the **best throughput** for the virtual memory system.

Swap-Space Management

Swap space use

- systems that implement swapping may use swap space to **hold an entire process image**, including the code and data segments.

- Paging systems may **simply store pages** that have been pushed out of main memory.

- The amount of swap space needed on a system can therefore vary from a few **megabytes of disk space to gigabytes**,
  - depending on the amount of physical memory,
  - the amount of virtual memory it is backing,
  - and the way in which the virtual memory is used.

Swap-Space Management

Swap space use

- Note that it may be safer to **overestimate than to underestimate** the amount of swap space required, because if a system **runs out of swap space it may be forced to abort processes or may crash entirely.**

- **Overestimation wastes disk space** that could otherwise be used for files, but it does no other harm.

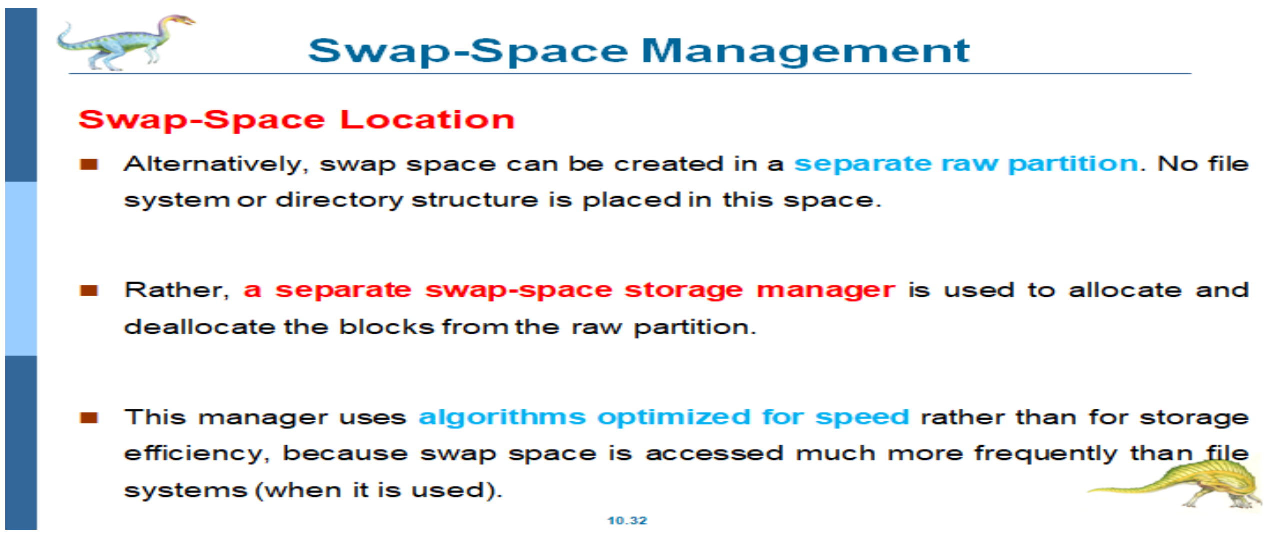
Swap-Space Management

Swap-Space Location

A swap space can reside in one of **two places**: it can be carved out of the **normal file system**, or it can be in a separate **disk partition**.

If the swap space is **simply a large file within the file system**, normal file-system routines can be used to create it, name it, and allocate its space.

This approach, though easy to implement, **is inefficient**.

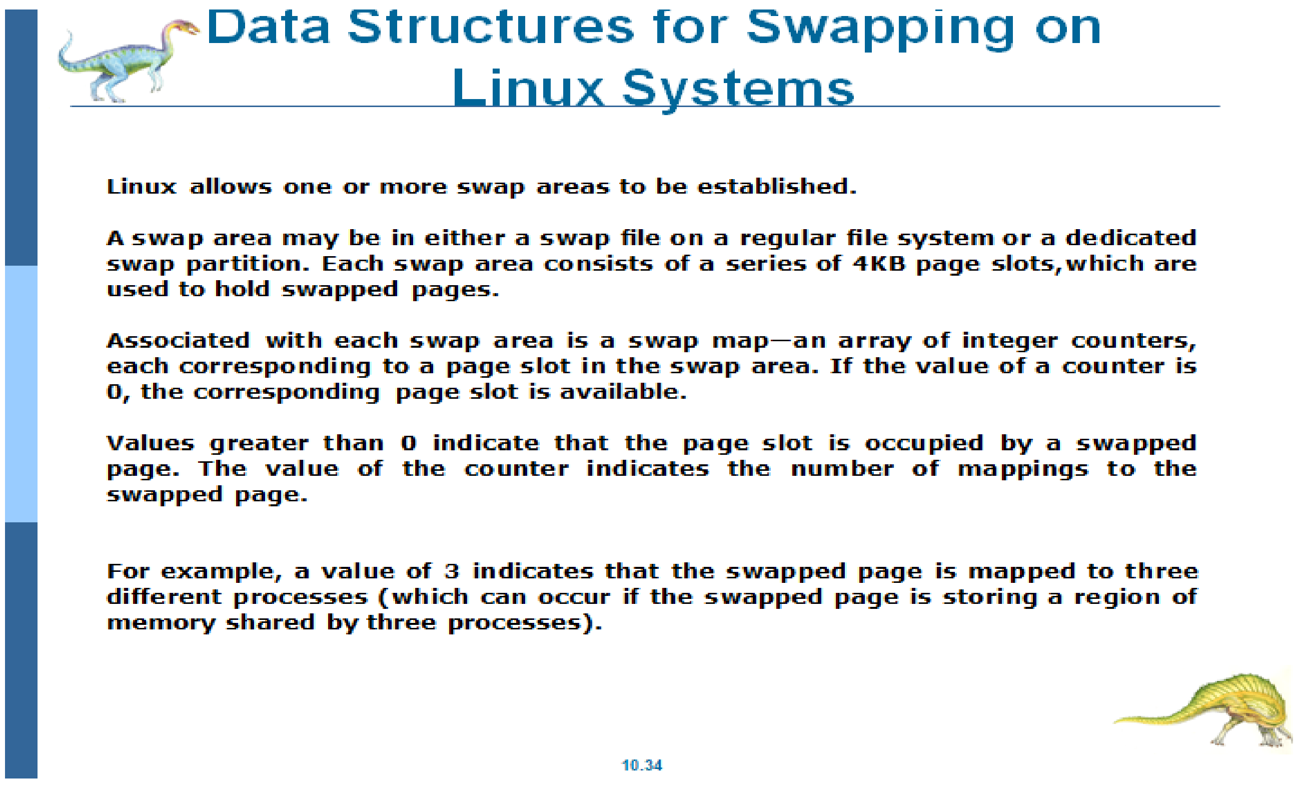


Swap-Space Management

Swap-Space Location

- Alternatively, swap space can be created in a separate raw partition. No file system or directory structure is placed in this space.
- Rather, a separate swap-space storage manager is used to allocate and deallocate the blocks from the raw partition.
- This manager uses algorithms optimized for speed rather than for storage efficiency, because swap space is accessed much more frequently than file systems (when it is used).

10.32



Data Structures for Swapping on Linux Systems

Linux allows one or more swap areas to be established.

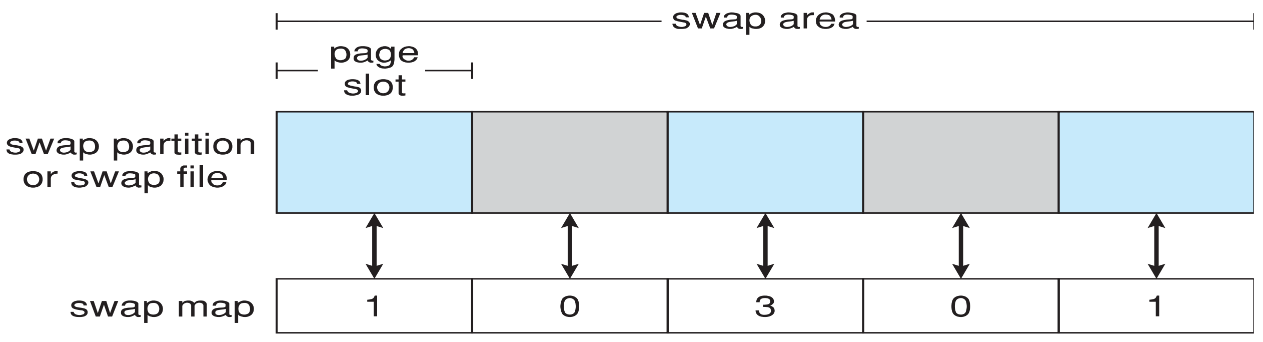
A swap area may be in either a swap file on a regular file system or a dedicated swap partition. Each swap area consists of a series of 4KB page slots, which are used to hold swapped pages.

Associated with each swap area is a swap map—an array of integer counters, each corresponding to a page slot in the swap area. If the value of a counter is 0, the corresponding page slot is available.

Values greater than 0 indicate that the page slot is occupied by a swapped page. The value of the counter indicates the number of mappings to the swapped page.

For example, a value of 3 indicates that the swapped page is mapped to three different processes (which can occur if the swapped page is storing a region of memory shared by three processes).

10.34



File Concept

- OS provides a uniform logical view (file) of various information storage devices by abstracting from the physical properties of storage devices
- **A file is a collection of related information that is recorded on secondary storage**
- Types of file
    - Data

      Numeric, alphabetic, alphanumeric, or binary
    - Program

      Source, object, executable

File Structure

- None - sequence of words, bytes such as text file
- Simple record structure
    - Lines
    - Fixed length record
    - Variable length record
- Complex structures
    - Formatted document (e.g. html)
    - Relocatable load file (e.g. object file)
- Can simulate last two with first method by inserting appropriate control characters

File Attributes

- A file has several attributes: (vary from one OS to another)
    - Name – only information kept in human-readable form
    - Identifier – unique tag (number) identifies file within file system
    - Type – needed for systems that support different types
    - Location – pointer to file location on device
    - Size – current file size
    - Protection – controls who can do reading, writing, executing
    - Time, date, and user identification – data for protection, security, and usage monitoring
- Information about files are kept in the directory structure, which is maintained on the disk

File Operations

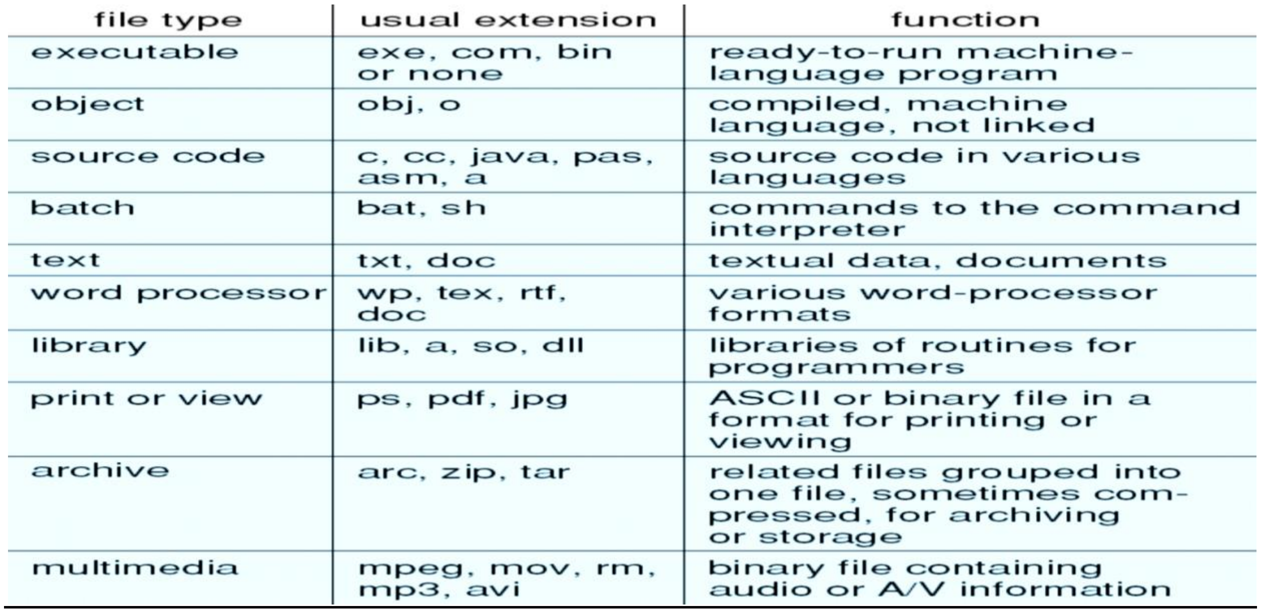
- OS provides various system calls for file operations
- For example:
    - Create
    - Write
    - Read
    - Reposition within file – file seek

- Delete
- Truncate
- Open(Fi) – search the directory structure on disk for entry Fi, and

move the content of entry to memory

- Close (Fi) – move the content of entry Fi in memory to directory

File Types – Name, Extension



file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

Access Methods

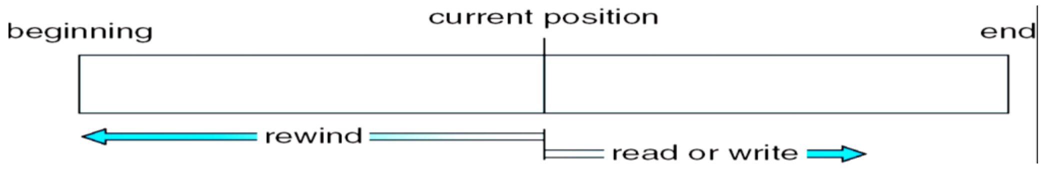
- **Sequential Access**: information in the file is processed in order
  - **read next**-reads the **next portion of the file** and automatically advances a file pointer
  - **write next** – **appends to the end of the file** and advances to the end of the newly written material (the new end of file).
  - **reset- file can be reset to the beginning,** and on some systems, a program may be able to skip forward or backward n records

    read next

write next

reset

skip forward or backward n records



- ❖ **Direct Access**: allow arbitrary blocks to be read or written

read n (n = relative block number)

write n

position to n

read next

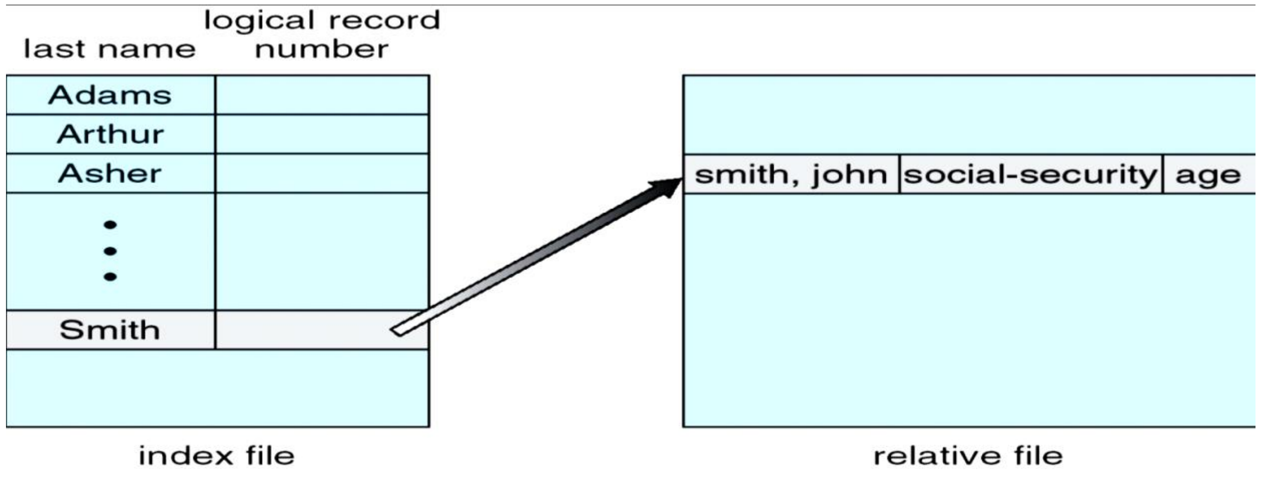
write next

rewrite n

<u>Simulation of Sequential Access on a Direct-access File</u>

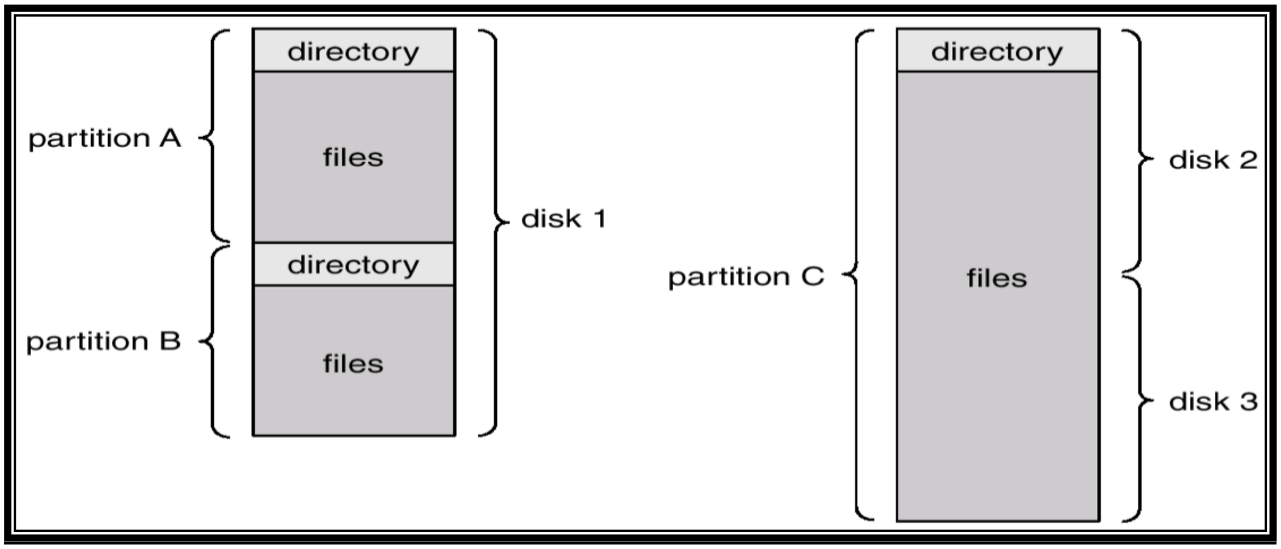
sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp* + 1;
write next	*write cp*; *cp* = *cp* + 1;

<u>Example of Index and Relative Files</u>



<u>Directory Structure</u>

<u>A Typical File-system Organization</u>



- Every partition has a file system, which consists of directory and files
- A collection of nodes containing information about all files
- Both the directory structure and the files reside on disk
- Backups of these two structures are kept on tapes

Information in a Directory

- Name
- Type
- Address
- Current length
- Maximum length
- Date last accessed
- Date last updated
- Owner ID
- Protection information

Operations Performed on a Directory

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

How to Organize a Directory?

- **Issues**
  - **Efficiency** – locating a file quickly
  - **Naming** – convenient to users

- ❖ Two users can have same name for different files
- ❖ The same file can have several different names

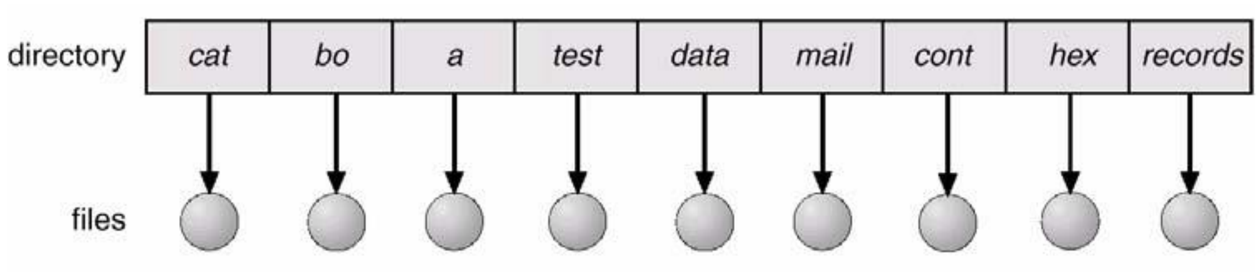
❖ **Grouping** – logical grouping of files by properties (e.g. all Java programs, all games, …)

❖ **Schemes for defining the logical structure of a directory:**

- ❖ Single level directory
- ❖ Two level directory
- ❖ Tree structured directory
- ❖ Acyclic graph directory
- ❖ General graph directory

Single-Level Directory

❖ A single directory for all users; easy to support and understand



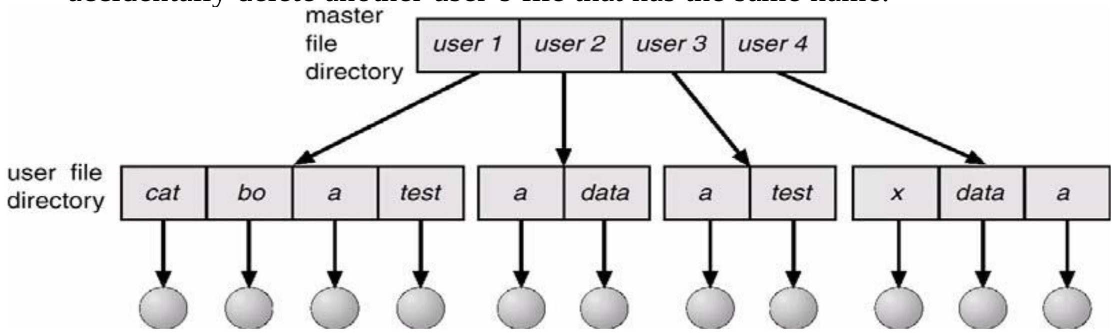
❖ Problems

- ❖ Naming problem: all files must have unique names
- ❖ Grouping problem

Two-Level Directory

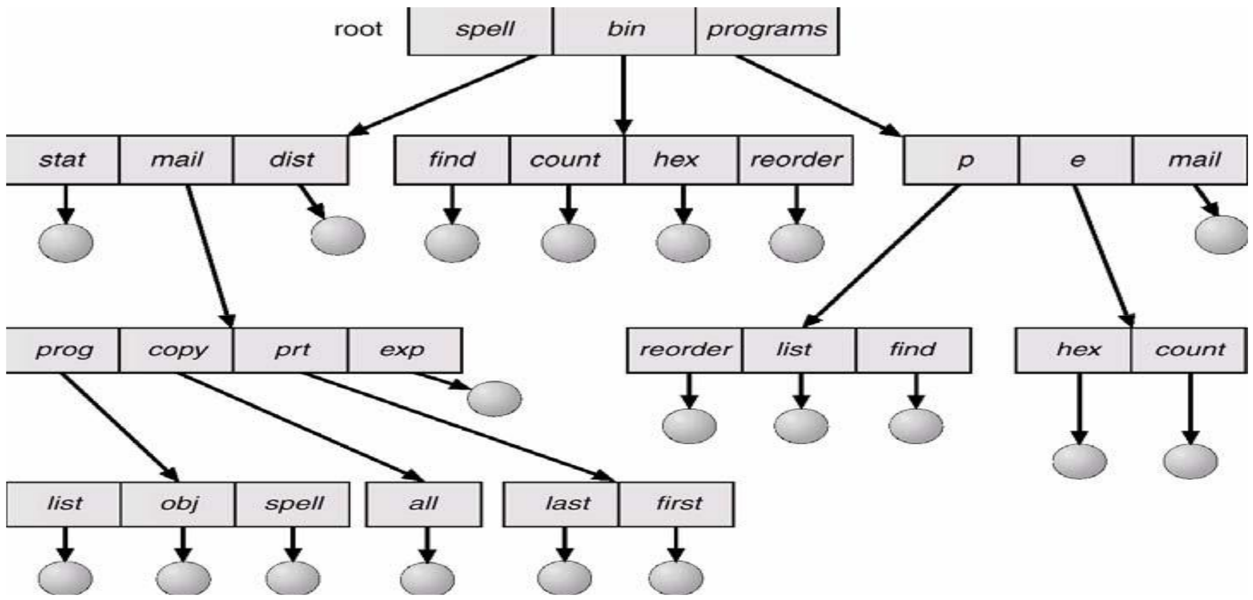
As, a single-level directory often leads to confusion of file names among different users.

- The standard solution is to create a separate directory for each user.
- In the two-level directory structure, each user has 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.
- When a user refers to a particular file, only his own UFD is searched.
- Thus, different users may have files with the same name, as long as all the file names within each UFD are unique.
- To create a file for a user, the operating system searches only that user's UFD to ascertain whether another file of that name exists.
- To delete a file, the operating system confines its search to the local UFD; thus, it cannot accidentally delete another user's file that has the same name.



- A user name and a file name define a path name
- Advantages and disadvantages
    - Can have the same file name for different user
    - Efficient searching
    - No grouping capability

Tree-Structured Directories



- 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 (or subdirectory) contains a set of files or subdirectories.
- All directories have the same internal format. One bit in each directory entry defines the entry as a file (0) or as a subdirectory (1).
- Special system calls are used to create and delete directories.
- 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 a path name or change the current directory to be the directory holding that file.
- To change directories, a system call is provided that takes a directory name as a parameter and uses it to redefine the current directory.
- Path names can be of two types: absolute and relative.

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

A **relative path name** defines a path from the current directory

Advantages

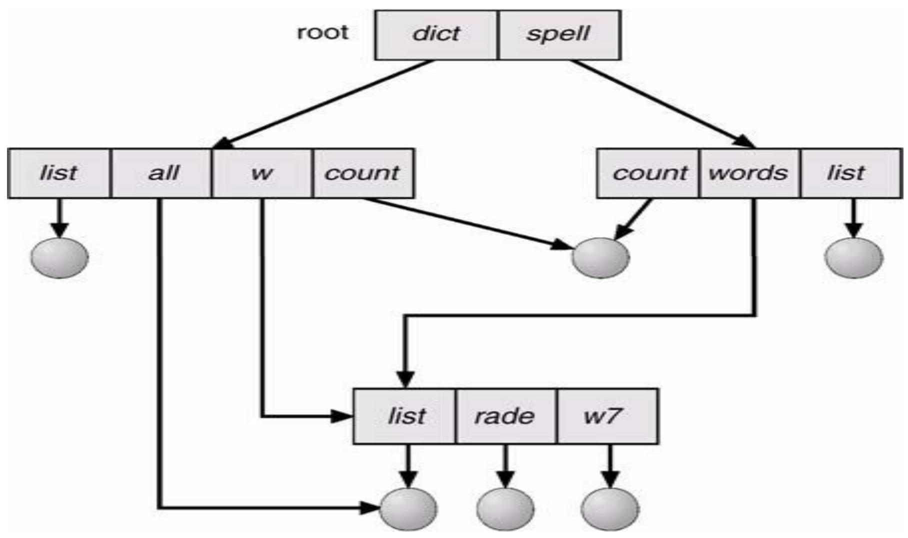
- ❖ Efficient searching
- ❖ Grouping Capability

DisAdvantages

- ❖ Subdirectories cannot be shared by multiple user

Acyclic-Graph Directories

- ❖ Have shared subdirectories and files using acyclic graph
- ❖ With a shared file **only one actual file exists**, so any changes made by one person are immediately visible to the other
- ❖ **Deletion:**
  - The deletion of a link does not need to affect the original file; only the link is removed.
  - Another approach to deletion is to preserve the file until all references to it are deleted.
  - Another approach keep a count of the number of references.
  - When count=0, file is deleted.



Allocation Methods

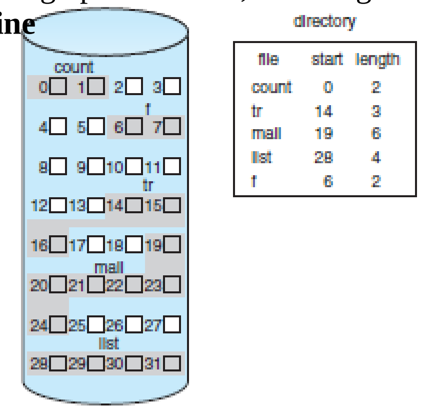
Three major methods of allocating disk space are in wide use**: contiguous, linked, and indexed**.

a) Contiguous Allocation

- **Contiguous allocation** requires that each file occupy a set of contiguous blocks on the disk.
- Disk addresses define a linear ordering on the disk. With this ordering, assuming that only one job is accessing the disk, accessing block *b* +1 after block *b* normally requires no head movement.
  - Simple – only starting location (block #) and length (number of blocks) are required

- Contiguous allocation has some problems, however. One difficulty is finding space for a new file.

- Problems include finding space for file, knowing file size, external fragmentation, need for **compaction off-line**



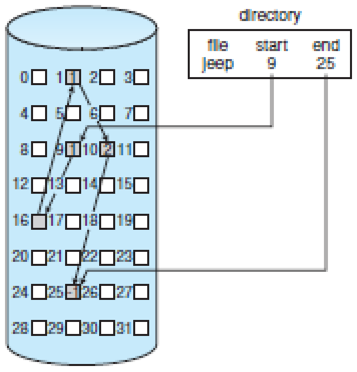
<u>**Extent-Based Systems**</u>

- Many newer file systems (i.e., Veritas File System) use a modified contiguous allocation scheme
- Extent-based file systems allocate disk blocks in extents
- An extent is a contiguous block of disks
  - Extents are allocated for file allocation
  - A file consists of one or more extents

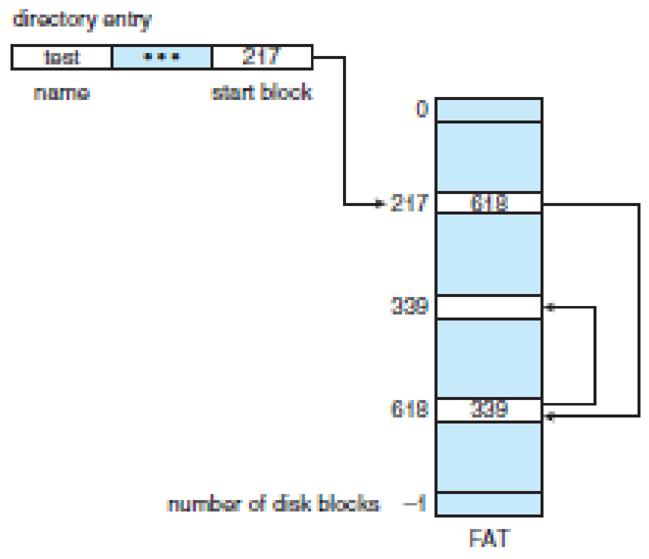
Linked allocation – each file a linked list of blocks

- File ends at nil pointer
- No external fragmentation
- Each block contains pointer to next block
- No compaction, external fragmentation
- Free space management system called when new block needed
- Improve efficiency by clustering blocks into groups but increases internal fragmentation
- Reliability can be a problem
- Locating a block can take many I/Os and disk seeks

■

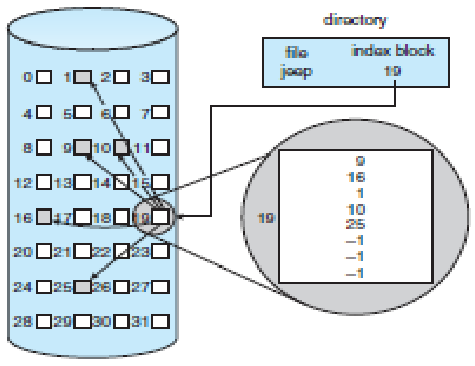


- An important variation on linked allocation is the use of a file-allocation table (FAT).
- The FAT is used in much the same way as a linked list. The directory entry contains the block number of the first block of the file. The table entry indexed by that block number contains the block number of the next block in the file.



c) Indexed Allocation

- Linked allocation solves the external-fragmentation and size-declaration problems of contiguous allocation
- Linked allocation cannot support efficient direct access, since the pointers to the blocks are scattered with the blocks themselves all over the disk and must be retrieved in order.
- Indexed allocation solves this problem by bringing all the pointers together into one location: the index block.
- Each file has its own index block, which is an array of disk-block addresses. The ith entry in the index block points to the ith block of the file.
- The directory contains the address of the index block .To find and read the ithblock, we use the pointer in the ith index-block entry.
- When the file is created, all pointers in the index block are set to null.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.



- If the index block is too small, however, it will not be able to hold enough pointers for a large file, and a mechanism will have to be available to deal with this issue.
- **Mechanisms for this purpose include the following:**

• **Linked scheme** - An index block is normally one disk block. Thus, it can be read and written directly by itself. To allow for large files, we can link together several index blocks.

• **Multilevel index** – A variant of linked representation uses a first-level index block to point to a set of second-level index blocks, which in turn point to the file blocks. To access a block, the operating system uses the first-level index to find a second-level index block and then uses that block to find the desired data block.

Part-A

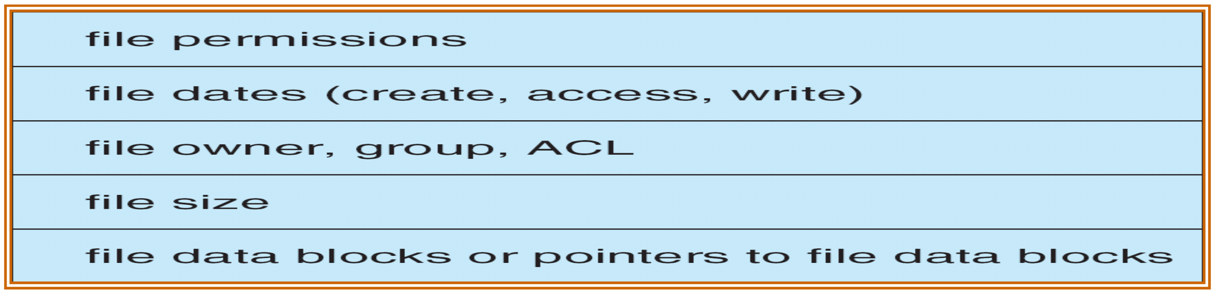
1.Give the disadvantages of Contiguous allocation

- ❖ Suffers from external fragmentation
- ❖ Suffers from internal fragmentation
- ❖ Difficulty in finding space for a new file
- ❖ File cannot be extended
- ❖ Size of the file is to be declared in advance

2.List the drawbacks of Linked Allocation of Disk Space.

- Cannot be used for random access – only sequential access.
- We must follow the pointers until we find the desired block.
- Not efficient if we need a direct-access capability.
- Also pointers do take up some space, if one adds them up!

3.List the content of FCB



file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks or pointers to file data blocks

4.Differentiate sector sparing and sector slipping?

Sector sparing:

Low-level formatting also sets aside spare sectors not visible to the operating system. The controller can be told to replace each bad sector logically with one of the spare sectors. This scheme is known as sector sparing or forwarding.

Sector slipping:

Sector slipping is a technique used to deal with defective **sectors**in hard disk drives. Due to the volatility of hard disks from their moving parts and low tolerances some **sectors** become defective.

5. **Give the importance of swap space management**

- Virtual memory uses swap space as an extension of main memory.
- Main goal for the design and implementation of swap space is to provide the best throughput for VM system.
- It uses to hold entire process image

6. **What is meant by absolute and relative path name?**

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

7.Explain seek time?

A disk is divided into many circular tracks.**Seek Time** refers to how long it takes the *read/ write head* on a hard disk to move from one track to another.

8.Explain rotational latency?

A **rotational Latency** is the time between information requests and how long it takes the hard drive to move to the correct sector.

9.Explain bandwidth?

Bandwidth is the total no of bytes transferred directly between the first request for service and the completion of last transfer.

10. **Define UFD and MFD**

In the two-level directory structure, each user has her own user file directory (UFD). Each UFD has a similar structure, but lists only the files of a single user. When a job starts the system's master file directory (MFD) is searched. The MFD is indexed by the user name or account number, and each entry points to the UFD for that user.

11.What is meant by Mounting? Give its advantages.

i. A file must be opened before it is used, a file system must be mounted before it can be available to processes on the system. The mount procedure is straight forward.

The operating system is given the name of the device, and the location within the file
Structure at which to attach the File system (or mount point)

12.How disk free space is managed using Bit vector? Give an example.

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.
For example , consider a disk where block 2,3,4,5,8,9,10,11,12,13,17,18,25,26and27
Are free, and the rest of the block are allocated. The free space bit map would be
001111001111110001100000011100000…