UNIT V:

FILE SYSTEM

CONTENTS

FILE CONCEPT

File Concept –

Computers can store information on various storage media, such as magnetic disks, magnetic tapes, and optical disks. These storage devices are usually nonvolatile.

A file is a named collection of related information that is recorded on secondary storage. Many different types of information may be stored in a file – source programs, object programs, numeric data, text, graphic images, sound recordings, and so on.

a) File Attributes –

A file is named, for the useful of its human users, and is referred to by its name. A name is usually a string of characters, such as sample.txt. Some systems differentiate between uppercase and lowercase characters in names, whereas other systems do not.

A file's attributes vary from one operating system to another but typically consist of these:

FILE CONCEPT

Name – The symbolic file name of the file.

Identifier – It is a unique number that identifies the file within the file system.

Type – This information is needed for systems that support different types of files.

Location – Location of the file on the device.

Size – The current size of the file.

Protection – Access control information determines who can do reading, writing or both.

Time and date – This information may be kept for creation and last modification.

FILE CONCEPT

b) File Operations –

The operating system can provide system calls to create, write, read, and delete files.

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.

Writing a file – To write a file, we make a system call specifying both the name of the file and the information to be written to the file.

Reading a file – To read from a file, we use a system call that specifies the name of the file and where (in memory) the next block of the file should be put.

Deleting a file – To delete a file, we search the directory for the named file. Having found the associated directory entry, we release all file space, so that it can be reused by other files, and erase the directory entry.

FILE CONCEPT

c) File Types –

The name is split into two parts – a name and an extension, usually separated by a period character.

Most operating systems allow users to specify file names as a sequence of characters followed by a period and terminated by an extension of additional characters. File name examples include resume.doc, server.java, and sample.txt.

The system uses the extension to indicate the type of the file. Only a file with a .com, .exe, or .bat extension can be executed.

The .com and .exe files are two forms of binary executable files, whereas a .bat file is a batch file containing, in ASCII format, commands to the operating system.

ACCESS METHODS

- 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.
- Some systems provide only one access method for files. Other systems, support many access methods.

a) Sequential Access –

b) Direct Access –

SEQUENTIAL ACCESS –

- The simplest access method is sequential access. Information in the file is processed in order, one record after the other. For example, editors and compilers usually access files in this fashion.
- A read operation – read next – reads the next portion of the file and automatically advances a file pointer, which tracks the I/O location. Similarly, the write operation - write next – appends to the end of the file.
- Such a file can be reset to the beginning; and on some systems, a program may be able to skip forward or backward ‘n’ records.
- Sequential access file, which is shown in following figure, is based on a tape model of a file and works as well on sequential access devices as it does on random access ones.

SEQUENTIAL ACCESS –

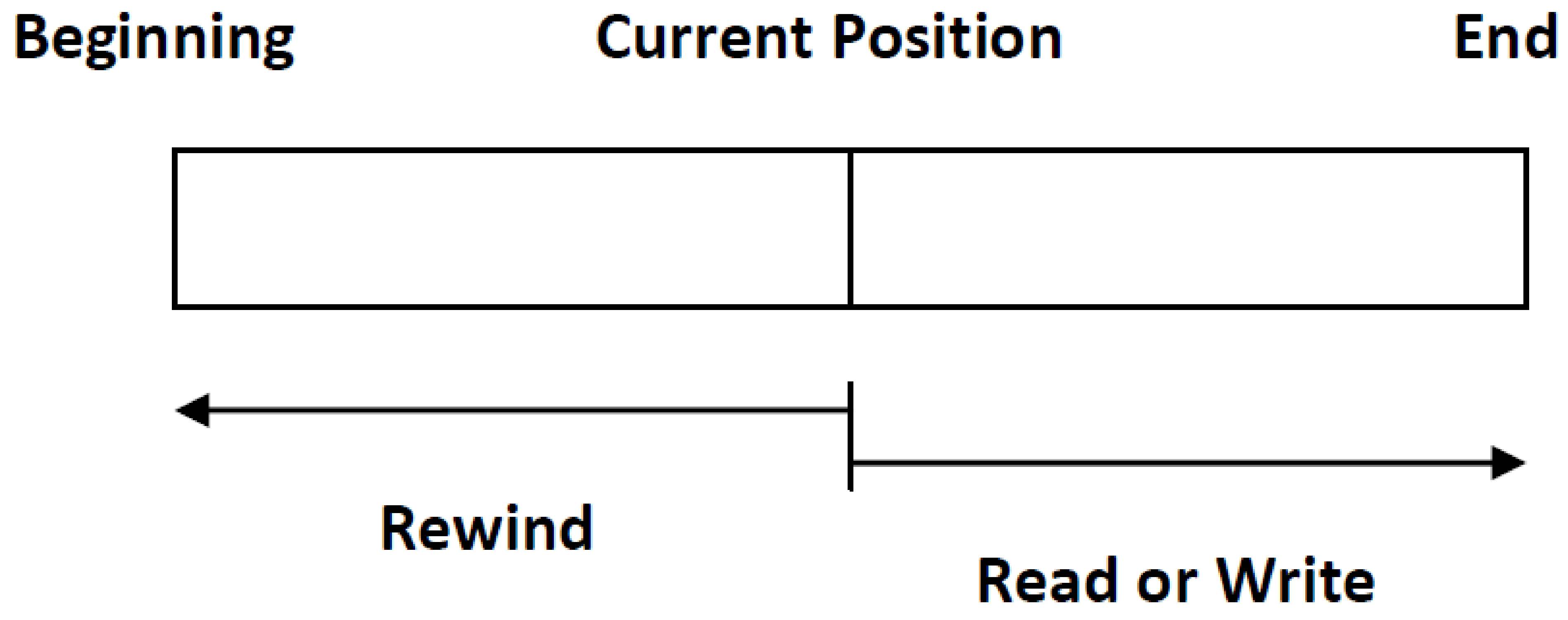


Figure – Sequential access file

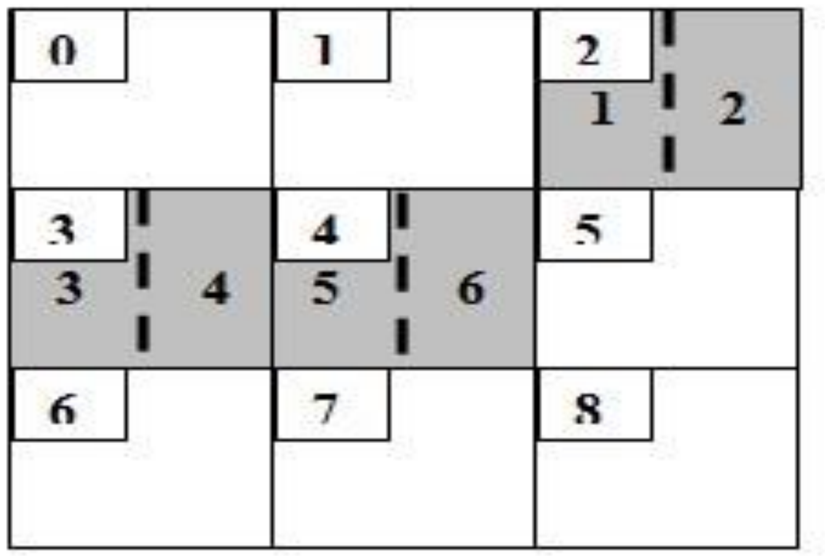
B) DIRECT ACCESS –

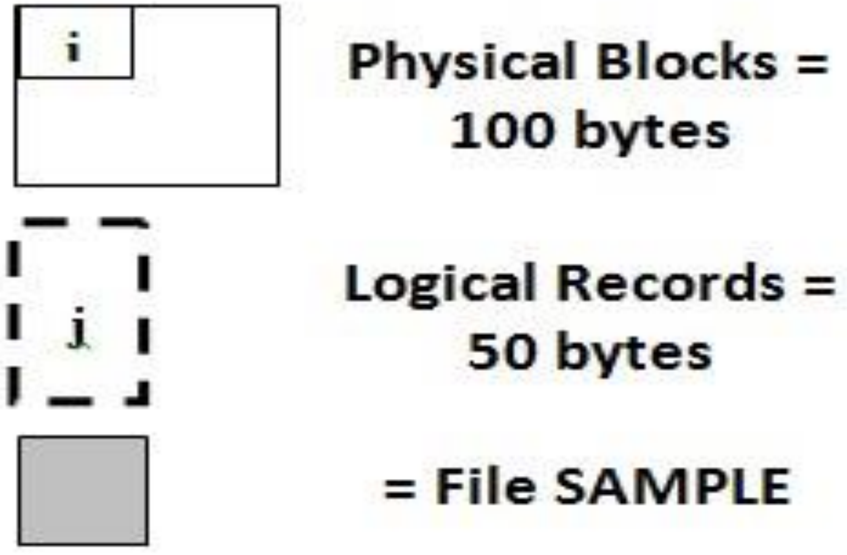
- Another method is direct access (or relative access). A file is made up of fixed length logical records that allow programs to read and write records rapidly in no particular order. The direct access method is based on a disk model of a file, since disks allow random access to any file block.
- We may read block 14, then read block 53, and then write block 7. There are no restrictions on the order of reading or writing for a direct access file. Direct access files are of great use for immediate access to large amounts of information.
- For the direct access method, the file operations must be modified to include the block number as a parameter. Thus, we have read 'n', where 'n' is the block number, rather than read next, and write 'n' rather than write next.
- Not all operating systems support both sequential and direct access for files. Some systems allow only sequential file access; others allow only direct access.

ACCESS METHODS

- Figure – Physical

  Block Storage





ACCESS METHODS

- Suppose, system wants to access record number 5 from the SAMPLE file. System identifies associated block number for that record using following formulas and then directly accessed the block and then accesses required record,
-
- Logical byte address = (Record Number - 1) * Record Length
- = (5 - 1) * 50
- = 200

ACCESS METHODS

Physical Block Number = Logical Byte Address /

Physical Block Size +

Address of First Physical Block

E.g. - Physical Block Number = 200 / 100 + 2

= 2 + 2

= 4

ACCESS METHODS

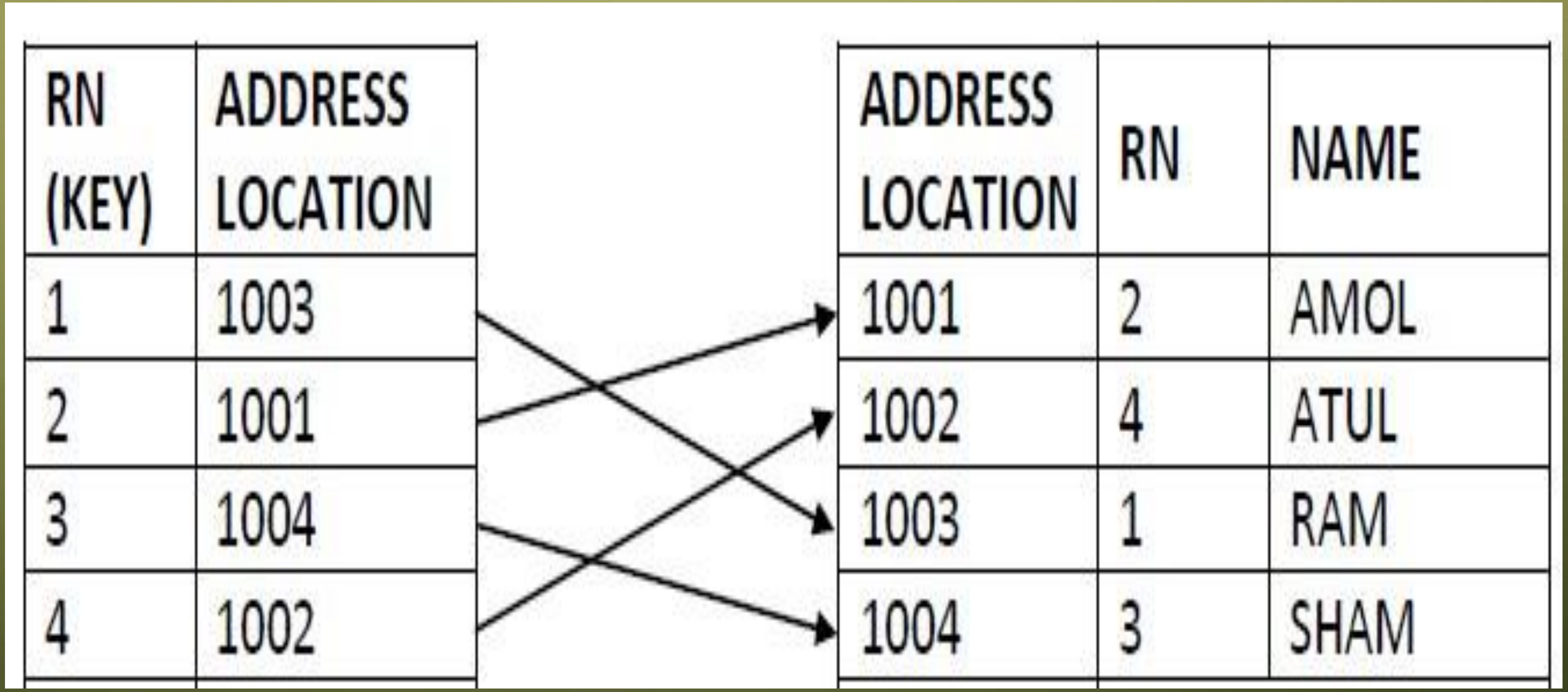
- This is the block (4) containing record 5 of file SAMPLE. There is no need to search in sequential order like 2, 3, and then 4. So, seeking time for direct access method is less as compared to the sequential access method

C) OTHER ACCESS METHODS –

- If we want to read specific section in book, we would not begin from page 1 & read every page until we across this section. Rather, we would search this section in the table of contents at the beginning of the book called as, 'INDEX'. Index file use same principle.
- Following figure shows, in indexed access method, there are two files for every data file i.e. MASTER file & INDEX file. MASTER file contains actual records in a file & INDEX file contains the index key & disk address of each record in the MASTER file.

C) OTHER ACCESS METHODS –

- Records in the MASTER file can be stored in random sequence, but index keys in the index file are stored in sorted sequence on index key value.



RN (KEY)	ADDRESS LOCATION
1	1003
2	1001
3	1004
4	1002

Table - Index File

ADDRESS LOCATION	RN	NAME
1001	2	AMOL
1002	4	ATUL
1003	1	RAM
1004	3	SHAM

Table - Master File

C) OTHER ACCESS METHODS –

- For processing a search request for a particular record, the computer first searches the index file to determine physical location of the record and then accesses the corresponding record from the data file.
- For example, to locate a student record who's RN (ROLL NUMBER) is 4. The computer searches index file first for this STUDENT – RN key & obtains the corresponding address value 1002. It then directly accesses the record stored at address location 1002 in the MASTER file.
-

C) OTHER ACCESS METHODS –

Advantage –

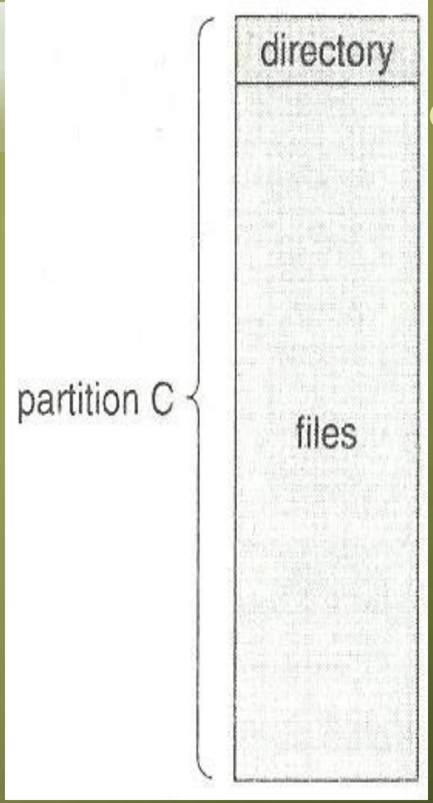
1) Records are accessed in very quickly.

Disadvantages –

It requires extra memory space to store index file.

If an index fails to operate, the whole system fails.

DIRECTORY STRUCTURE –



- A disk can be used in its entirety for a file system. A file system can be created on each of the parts of the disk.
- **Figure – A typical file system organization**
- Each volume that contains a file system must also contain information about the files in the system. This information is kept in entries in a device directory or volume table of contents. The device directory records information – such as name, location, size, and type for all files on that volume.

DIRECTORY STRUCTURE

- When considering a particular directory structure, we need to keep in mind the operations that are to be performed on a directory:
  - **Search for a file –** To find a particular file in the directory.
  - **Create a file –** After a new file is created, its entry will be added to the directory.
  - **Delete a file –** Remove the entry of deleted file from the directory.
  - **List a directory –** To see the list of available files.
  - **Rename a file –** We must be able to change the name as per requirement.

A) 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 as shown in following figure,

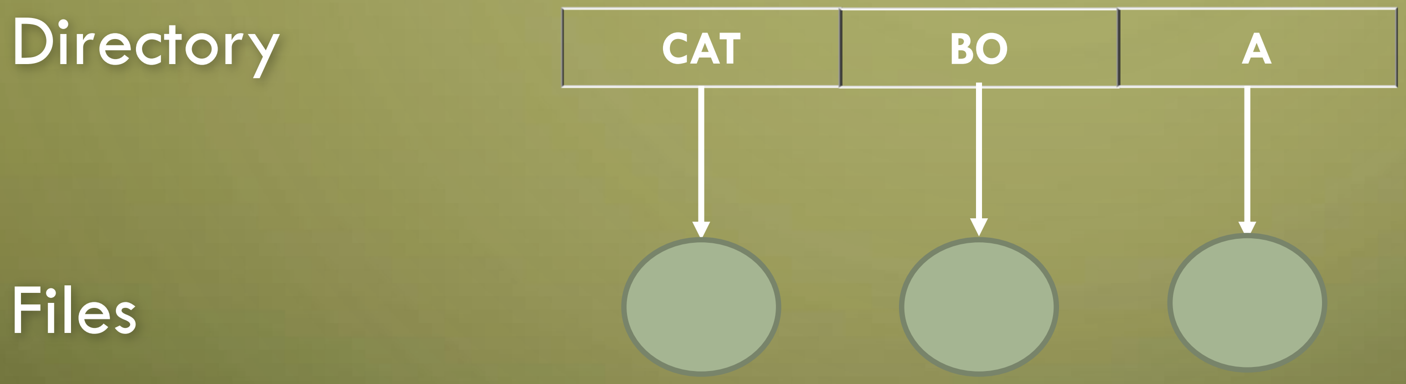


Figure – Single level directory

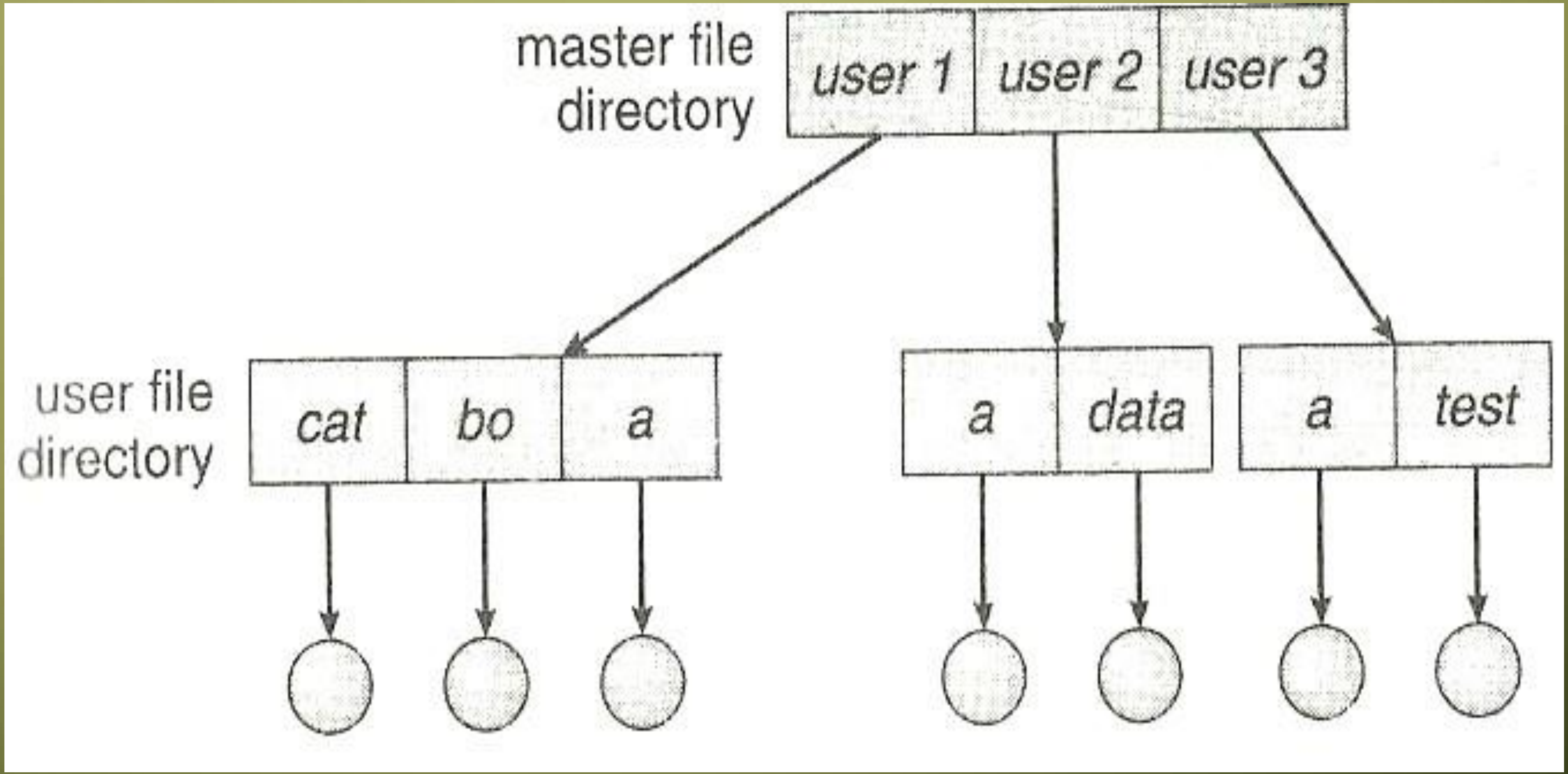
A) SINGLE LEVEL DIRECTORY –

- 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.
-

B) TWO LEVEL DIRECTORY –

- 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 structures, 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 logs in, the system's master file directory (MFD) is searched. The MFD is indexed by user name or account number, and each entry points to the UFD for that user as shown in following figure,

B) TWO LEVEL DIRECTORY –



- Figure – Two level directory structure
-

B) TWO LEVEL DIRECTORY –

- When a user refers to a particular file, only his own UFD is searched. Thus, different users may have files with the same name. To create a file for a user, the operating system searches only that user's UFD to check whether another file of that name exists. To delete a file, the operating system confines its search to the local UFD.
- A two level directory can be thought of as a tree, of height 2. The root of the tree is the MFD. Its direct descendants are the UFDs. The descendants of the UFDs are the files themselves.

C) TREE STRUCTURED DIRECTORIES

- Tree structured directories allows users to create their own subdirectories and to organize their files accordingly. 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. 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 entry defines the entry as a file (0) or as a subdirectory (1).

C) TREE STRUCTURED DIRECTORIES

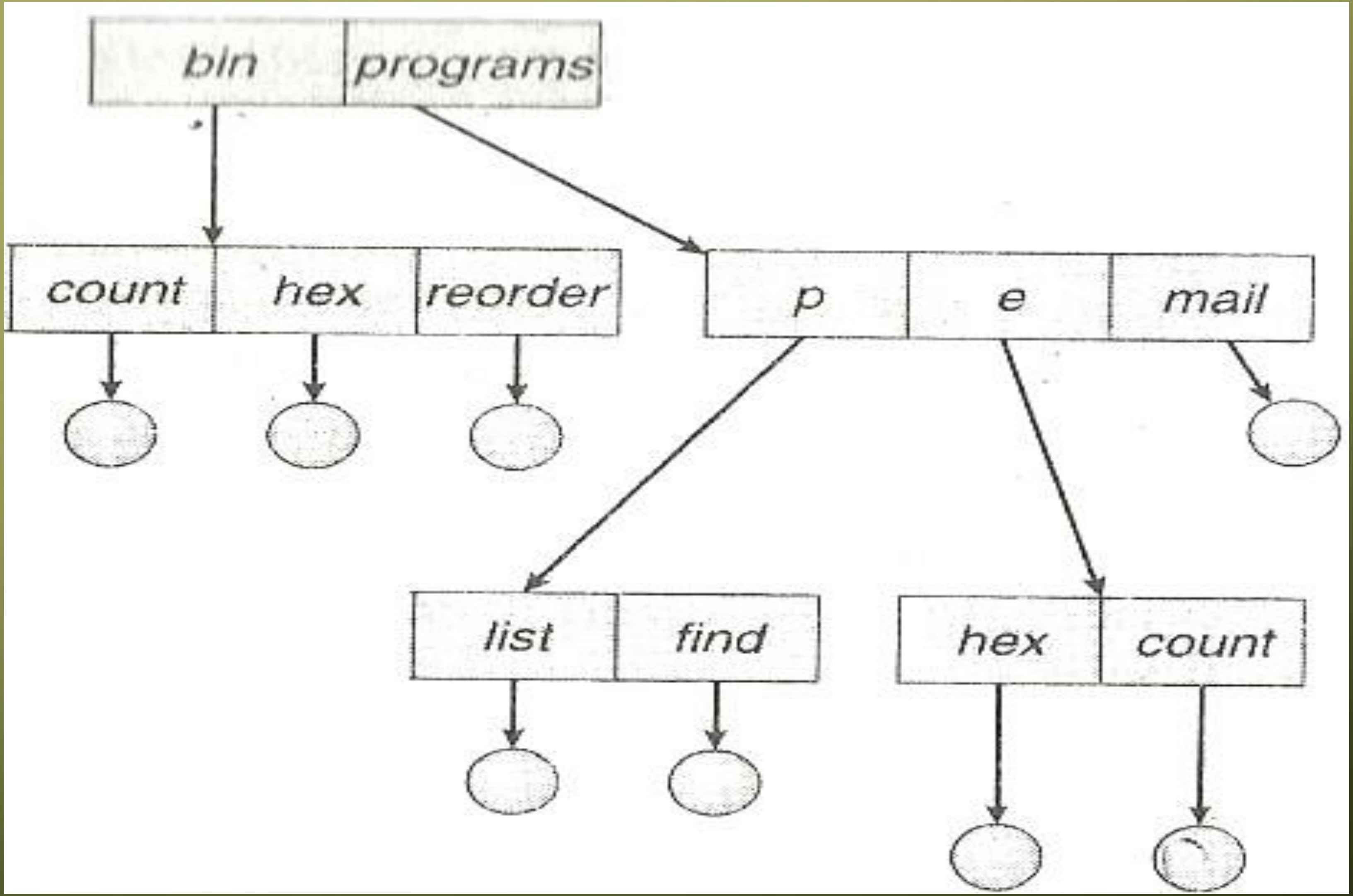


Figure – Tree structured directory structure

C) TREE STRUCTURED DIRECTORIES

- For example, in the tree structured file system of above figure, the relative path is p/list and the absolute path is programs/p/list for same file i.e. list file.
- If the directory to be deleted is not empty then one of two approaches can be taken. Some systems, such as MS-DOS, will not delete a directory unless it is empty. Thus, to delete a directory, the user must first delete all the files in that directory.
- An alternative approach, such as that taken by the UNIX is to provide an option: When a request is made to delete a directory that entire directory's files and sub directories are also to be deleted.

D) ACYCLIC GRAPH DIRECTORIES –

- 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 as shown in following figure.
- The common subdirectory should be shared. A shared directory or file will exist in the file system in two (or more) places at once.
- 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 original copy. 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.

D) ACYCLIC GRAPH DIRECTORIES –

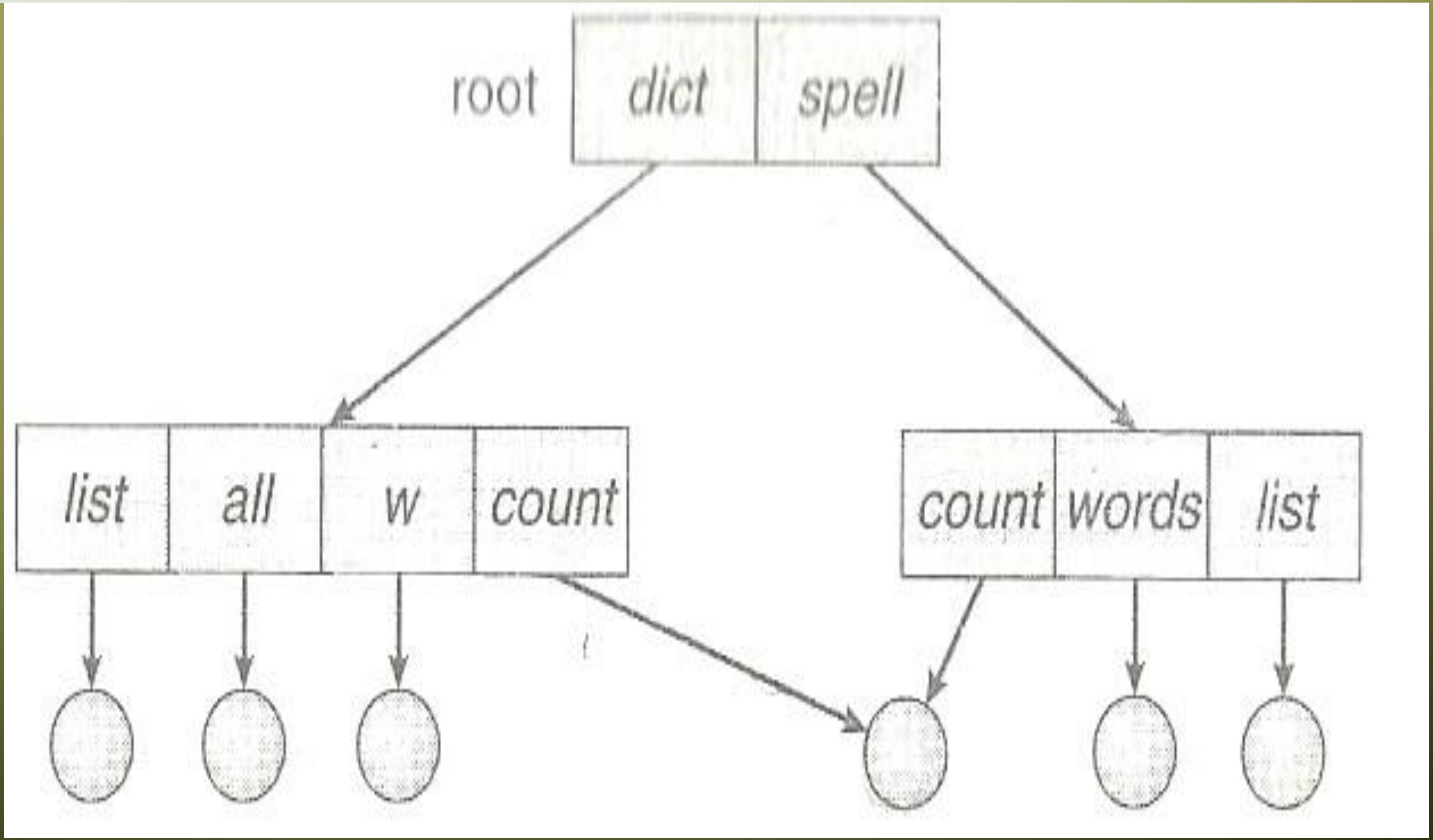


Figure – Acyclic graph directory structure

D) ACYCLIC GRAPH DIRECTORIES –

- Sharing is particularly important for subdirectories; a new file created by one person will automatically appear in all the shared subdirectories. An acyclic graph directory structure is more flexible than is a simple tree structure, but it is also more complex.

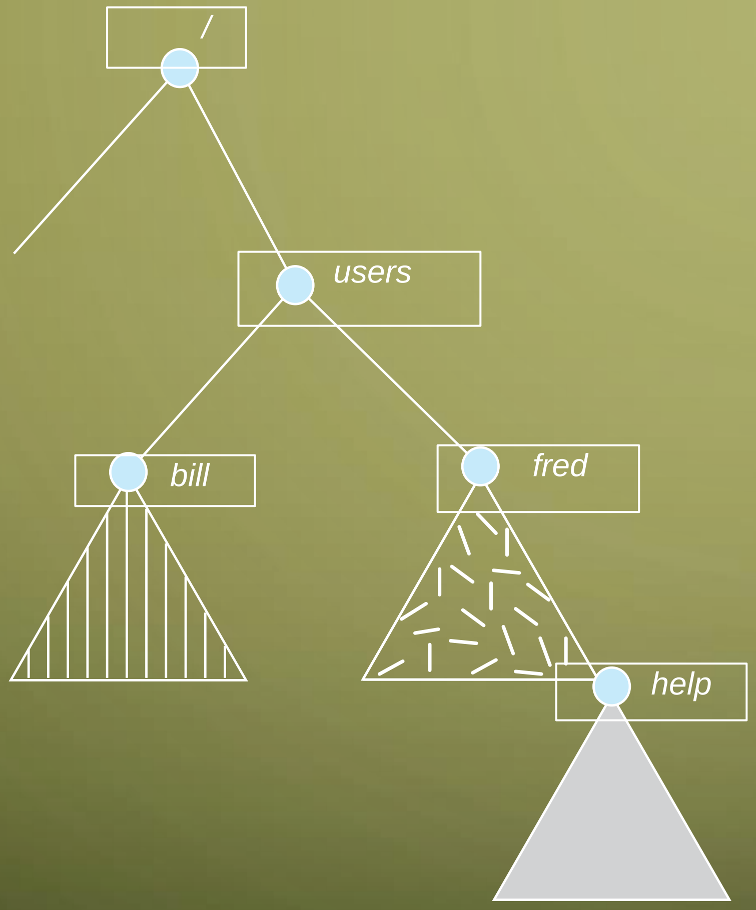
FILE-SYSTEM MOUNTING

- Just as 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.
- More specifically, the directory structure may be built out of multiple volumes, which must be mounted to make them available within the file-system name space.
- The mount procedure is straightforward. The operating system is given the name of the device and the mount point—the location within the file structure where the file system is to be attached.
- Some operating systems require that a file system type be provided, while others inspect the structures of the device and determine the type of file system.

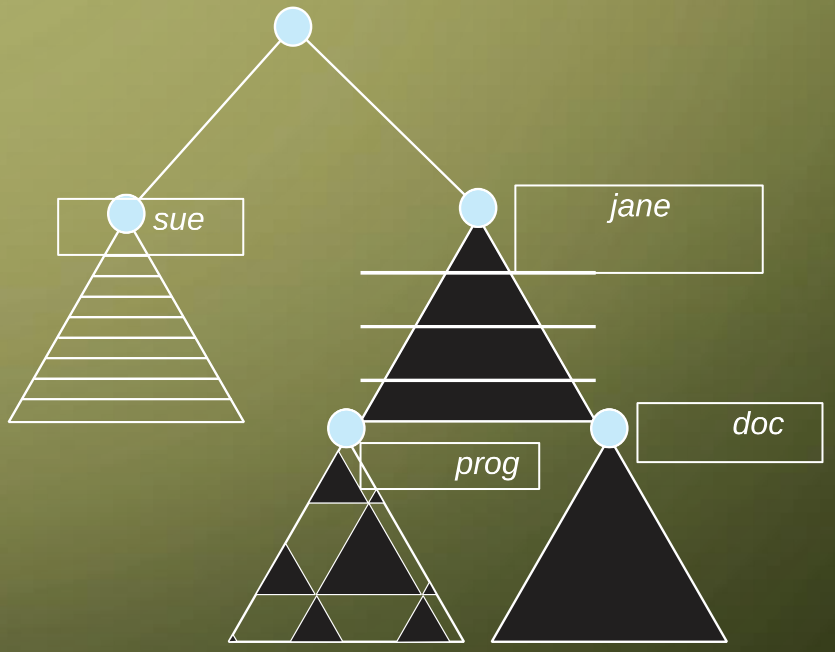
FILE-SYSTEM MOUNTING

- Typically, a mount point is an empty directory. For instance, on a UNIX system, a file system containing a user’s home directories might be mounted as /home; then, to access the directory structure within that file system, we could precede the directory names with /home, as in /home/jane. Mounting that file system under /users would result in the path name /users/jane, which we could use to reach the same directory.
- Next, the operating system verifies that the device contains a valid file system. It does so by asking the device driver to read the device directory and verifying that the directory has the expected format.
- Finally, the operating system notes in its directory structure that a file system is mounted at the specified mount point. This scheme enables the operating system to traverse its directory structure, switching among file systems, and even file systems of varying types, as appropriate.

FILE-SYSTEM MOUNTING



(a)



(b)

Figure File system. (a) Existing system. (b) Unmounted volume.

FILE-SYSTEM MOUNTING

- To illustrate file mounting, consider the file system depicted in above Figure where the triangles represent subtrees of directories that are of interest.
- Figure (a) shows an existing file system, while Figure (b) shows an unmounted volume residing on /device/dsk. At this point, only the files on the existing file system can be accessed.

FILE-SYSTEM MOUNTING

- The Microsoft Windows family of operating systems maintains an extended two-level directory structure, with devices and volumes assigned drive letters. Volumes have a general graph directory structure associated with the drive letter. The path to a specific file takes the form of drive-letter:\path\to\file.
- The more recent versions of Windows allow a file system to be mounted anywhere in the directory tree, just as UNIX does. Windows operating systems automatically discover all devices and mount all located file systems at boot time.

FILE SHARING

- Such file sharing is very desirable for users who want to collaborate and to reduce the effort required to achieve a computing goal. Therefore, user-oriented operating systems must accommodate the need to share files in spite of the inherent difficulties
- Multiple Users
- Remote File Systems

MULTIPLE USERS

- When an operating system accommodates multiple users, the issues of file sharing, file naming, and file protection become preeminent. Given a directory structure that allows files to be shared by users, the system must mediate the file sharing. The system can either allow a user to access the files of other users by default or require that a user specifically grant access to the files.

REMOTE FILE SYSTEMS

- Networking allows the sharing of resources spread across a campus or even around the world. One obvious resource to share is data in the form of files.
- Through the evolution of network and file technology, remote file-sharing methods have changed.
- The first implemented method involves manually transferring files between machines via programs like ftp.
- The second major method uses a **distributed file system (DFS)** in which remote directories are visible from a local machine. In some ways, the third method, the **WorldWide Web**, is a reversion to the first.

REMOTE FILE SYSTEMS

- A browser is needed to gain access to the remote files, and separate operations (essentially a wrapper for ftp) are used

  to transfer files.
- ftp is used for both anonymous and authenticated access. **Anonymous access** allows a user to transfer files without having an account on the remote system.

PROTECTION

- File owner/creator should be able to control:
  - what can be done
  - by whom
- Types of access
  - Read
  - Write
  - Execute
  - Append
  - Delete
  - List

PROTECTION

- When information is stored in a computer system, we want to keep it safe from physical damage and improper access
- Reliability is generally provided by duplicate copies of files. Many computers have systems programs that automatically copy disk files to tape at regular intervals to maintain a copy should a file system be accidentally destroyed.
- File systems can be damaged by hardware problems such as errors in reading or writing, power surges or failures, head crashes, dirt, temperature extremes, and vandalism.

PROTECTION

- Files may be deleted accidentally. Bugs in the file-system software can also cause file contents to be lost
- Protection can be provided in many ways. For a single-user laptop system, we might provide protection by locking the computer in a desk drawer or file cabinet. In a larger multiuser system, however, other mechanisms are needed.

PROTECTION

- 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
  - Access is permitted or denied depending on several factors, one of which is the type of access requested. Several different types of operations may be controlled:

PROTECTION

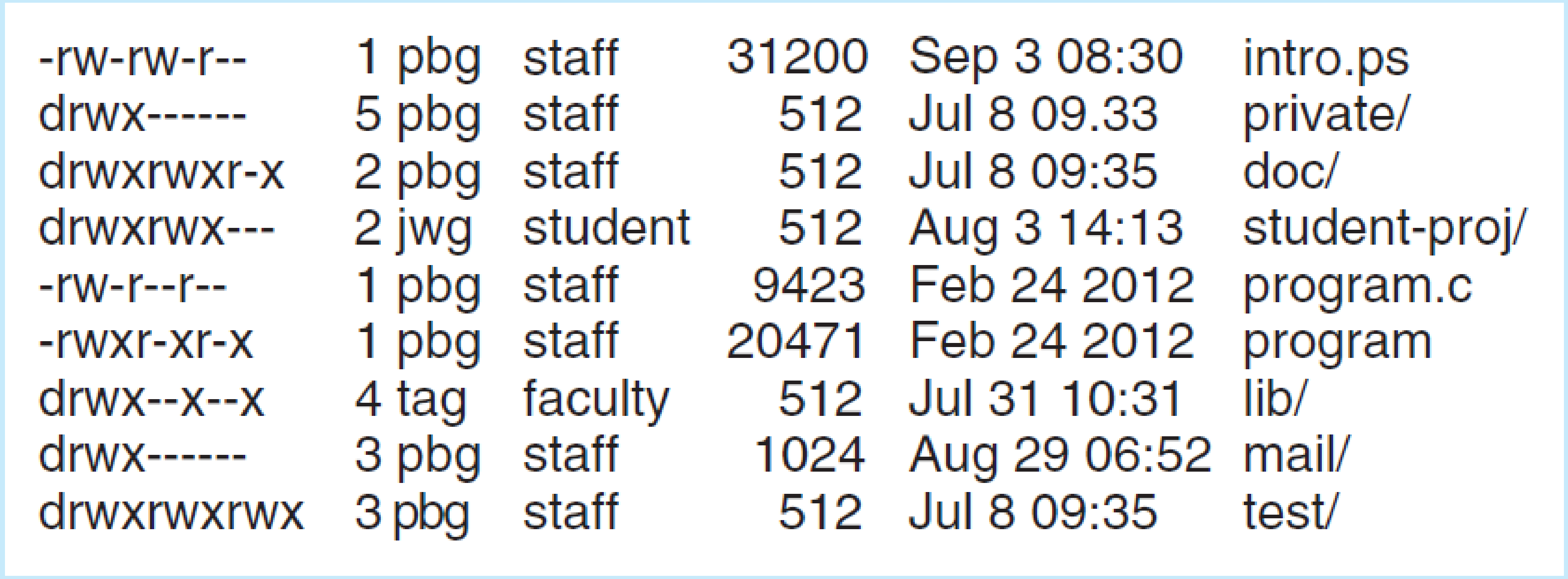
- **Read**. Read from the file.
- **Write.** Write or rewrite the file.
- **Execute.** Load the file into memory and execute it.
- **Append.** Write new information at the end of the file.
- **Delete.** Delete the file and free its space for possible reuse.
- **List.** List the name and attributes of the file.

PROTECTION

- Access Control:
  - **Access-control list (ACL)** specifying user names and the types of access allowed for each user. When a user requests access to a particular file, the operating system 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.
  - 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.

PROTECTION

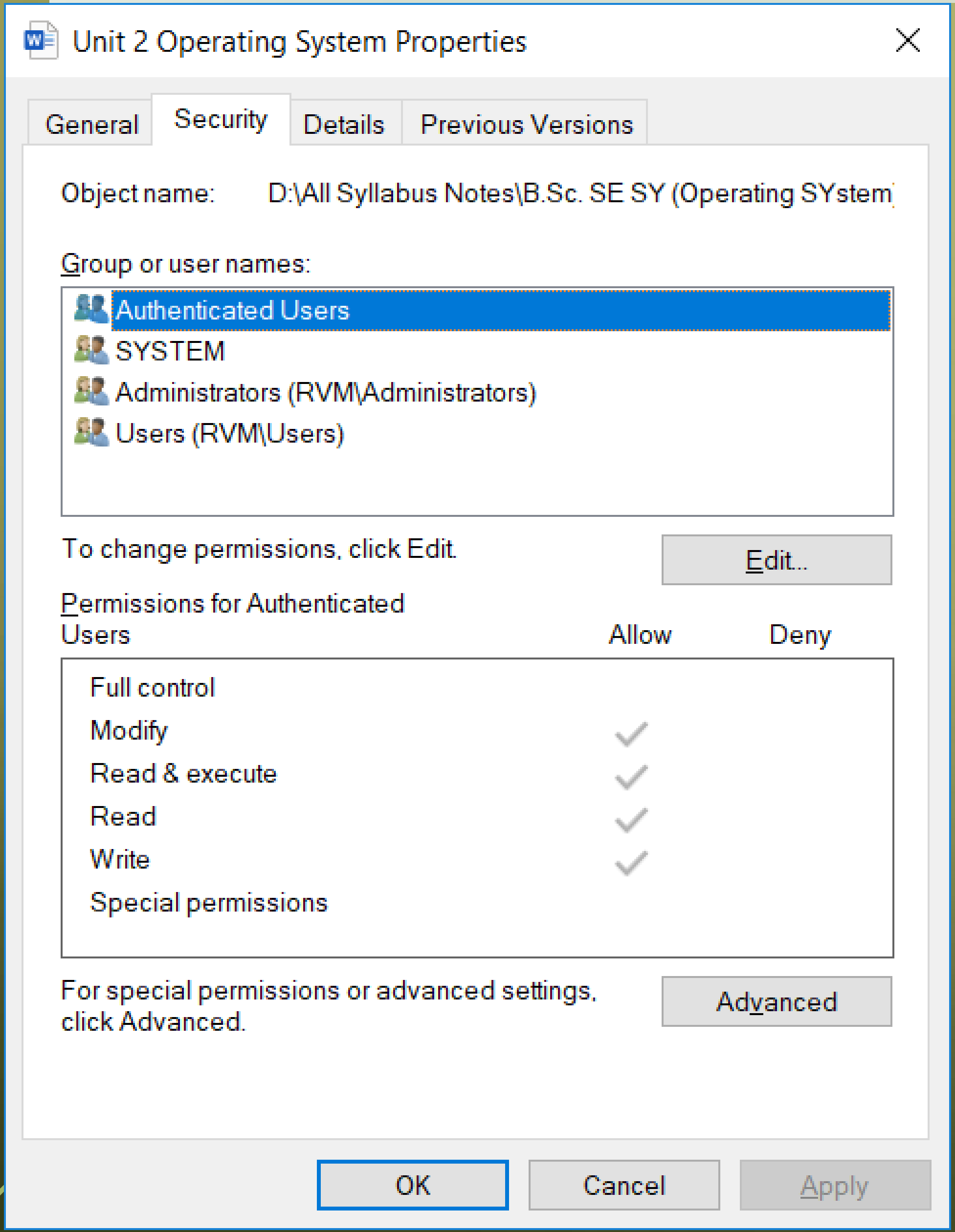
A sample directory listing from a UNIX environment is shown in below:



```
-rw-rw-r--     1 pbg   staff     31200  Sep 3 08:30    intro.ps
drwx------     5 pbg   staff       512  Jul 8 09.33    private/
drwxrwxr-x     2 pbg   staff       512  Jul 8 09:35    doc/
drwxrwx---     2 jwg   student     512  Aug 3 14:13    student-proj/
-rw-r--r--     1 pbg   staff      9423  Feb 24 2012    program.c
-rwxr-xr-x     1 pbg   staff     20471  Feb 24 2012    program
drwx--x--x     4 tag   faculty     512  Jul 31 10:31   lib/
drwx------     3 pbg   staff      1024  Aug 29 06:52   mail/
drwxrwxrwx     3 pbg   staff       512  Jul 8 09:35    test/
```

- R- Read, W- Write, X- Execute, d- Directory

PROTECTION



Unit 2 Operating System Properties

Object name: D:\All Syllabus Notes\B.Sc. SE SY (Operating SYstem

Group or user names:

- Authenticated Users
- SYSTEM
- Administrators (RVM\Administrators)
- Users (RVM\Users)

To change permissions, click Edit.

Permissions for Authenticated Users	Allow	Deny
Full control		
Modify	✓	
Read & execute	✓	
Read	✓	
Write	✓	
Special permissions		

For special permissions or advanced settings, click Advanced.

FILE-SYSTEM STRUCTURE

- Disks provide most of the secondary storage on which file systems are maintained. Two characteristics make them convenient for this purpose:
- 1. A disk can be rewritten in place; it is possible to read a block from the disk, modify the block, and write it back into the same place.
- 2. A disk can access directly any block of information it contains. Thus, it is simple to access any file either sequentially or randomly, and switching from one file to another requires only moving the read–write heads and waiting for the disk to rotate.
- To improve I/O efficiency, I/O transfers between memory and disk are performed in units of blocks. Each block has one or more sectors. Depending on the disk drive, sector size varies from 32 bytes to 4,096 bytes; the usual size is 512 bytes

FILE-SYSTEM STRUCTURE

- **File systems** provide efficient and convenient access to the disk by allowing data to be stored, located, and retrieved easily.
- A file system poses two quite different design problems. The first problem is **defining how the file system should look to the user**. This task involves defining a file and its attributes, the operations allowed on a file, and the directory structure for organizing files. The second problem is **creating algorithms and data structures to map the logical file system onto the** physical secondary-storage devices.
- The file system itself is generally composed of many different levels. The structure shown in Figure 12.1 is an example of a layered design. Each level in the design uses the features of lower levels to create new features for use by higher levels.

FILE-SYSTEM STRUCTURE

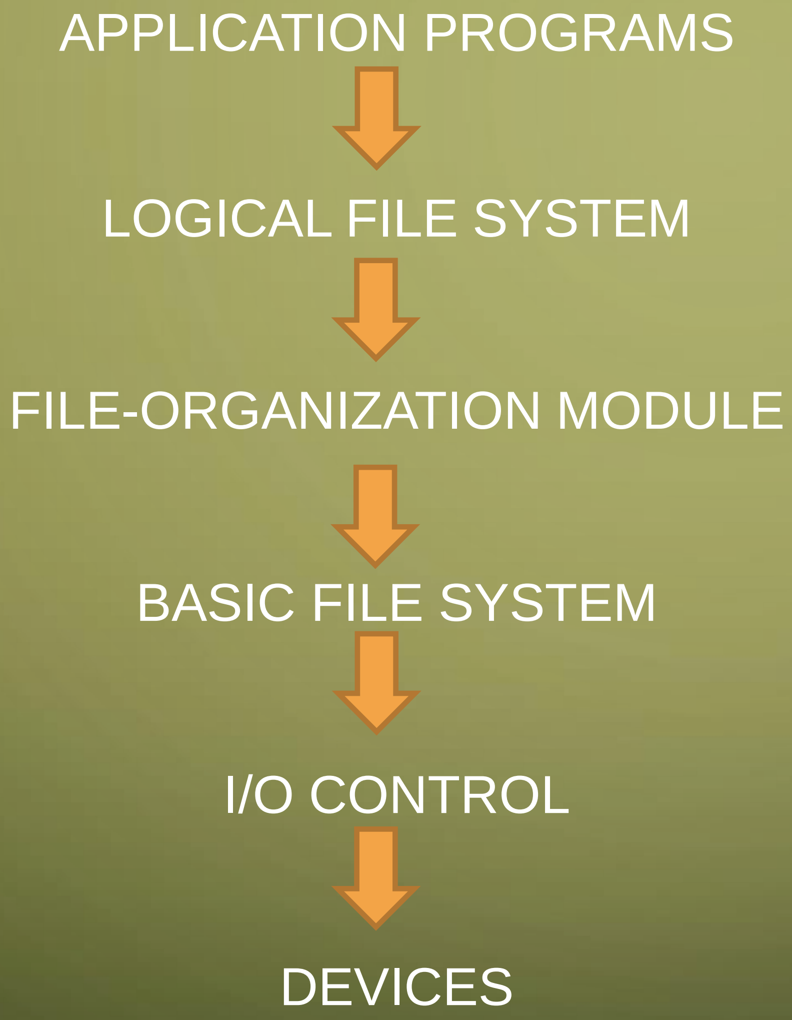


FIGURE: LAYERED FILE SYSTEM

FILE-SYSTEM STRUCTURE

- The **I/O control** level consists of device drivers and interrupt handlers to transfer information between the main memory and the disk system. A device driver can be thought of as a translator. Its input consists of highlevel commands such as “retrieve block 123.” Its output consists of low-level, hardware-specific instructions that are used by the hardware controller, which interfaces the I/O device to the rest of the system. The device driver usually writes specific bit patterns to special locations in the I/O controller’s memory to tell the controller which device location to act on and what actions to take.

- The **basic file system** needs only to issue generic commands to the appropriate device driver to read and write physical blocks on the disk. Each physical block is identified by its numeric disk address (for example, drive 1, cylinder 73, track 2, sector 10). This layer also manages the memory buffers and caches that hold various file-system, directory, and data blocks. A block in the buffer is allocated before the transfer of a disk block can occur. When the buffer is full, the buffer manager must find more buffer memory or freeup buffer space to allow a requested I/O to complete. Caches are used to hold frequently used file-system metadata to improve performance, so managing their contents is critical for optimum system performance.