4.5 DIRECTORY AND DISK STRUCTURE

- Directory & disk structure deals with how to store files. A disk can be **partitioned** into quarters, and each quarter can hold a separate file system.
- Partitioning is useful for limiting the sizes of individual file systems, putting multiple file-system types on the same device.
- Any entity containing a file system is generally known as a **volume**.
- 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 (more commonly known simply as the **directory**) records information—such as name, location, size, and type—for all files on that volume.

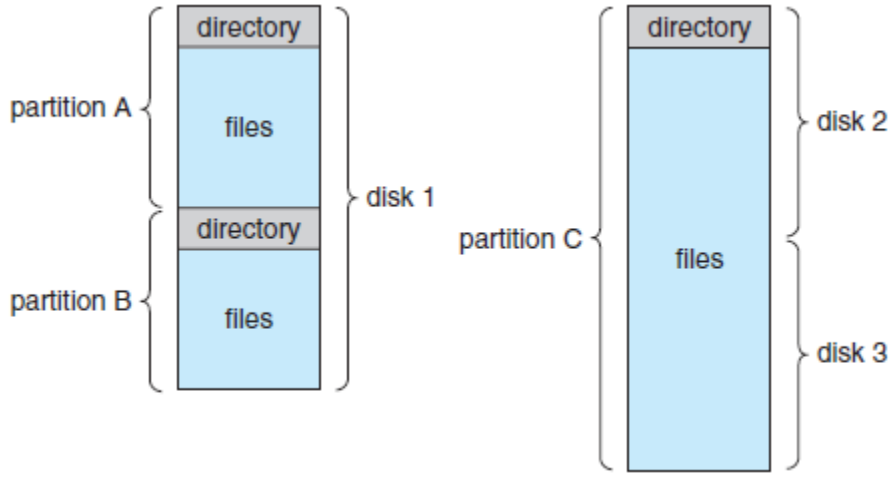


Fig: A typical file-system organization

Storage Structure

- A general-purpose computer system has multiple storage devices, and those devices can be sliced up into volumes that hold file systems.
- Computer systems may have zero or more file systems, and the file systems may be of varying types.
- For example, a typical Solaris system may have dozens of file systems of a dozen different types,

Consider the types of file systems in the Solaris

• **tmpfs**—a "temporary" file system that is created in volatile main memory and has its contents erased if the system reboots or crashes

• **lofs**—a "loop back" file system that allows one file system to be accessed in place of another one

• **procfs**—a virtual file system that presents information on all processes as a file system

• **ufs, zfs**—general-purpose file systems

Directory Overview

Directory operations to be supported include:

• Search for a file

• Create a file - add to the directory

• Delete a file - erase from the directory

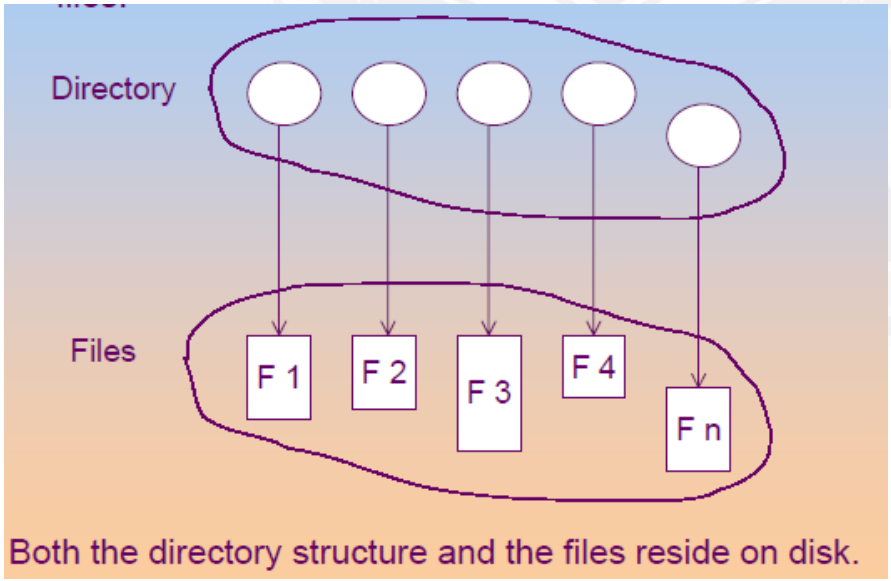
• List a directory - possibly ordered in different ways.

• Rename a file - may change sorting order

• Traverse the file system.

Directory Structure

A collection of nodes containing information about all files.



There are five directory structures. They are

1. Single-level directory

2. Two-level directory

3. Tree-Structured directory

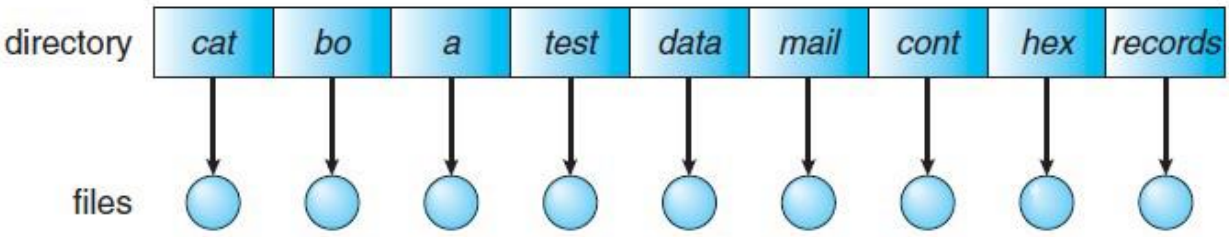
4. Acyclic Graph directory

5. General Graph directory

1. Single – Level Directory

- The simplest directory structure is the single- level directory.
- All files are contained in the same directory.
- Disadvantage:

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.

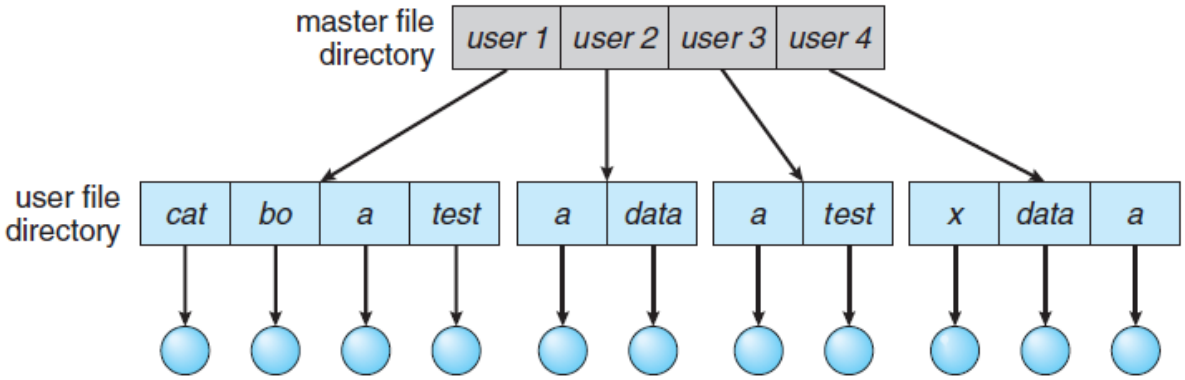


2. Two – Level Directory

- Separate directory for each user.
- In the two level directory structures, each user has her own user file directory (UFD).
- When a user job starts or a user logs in, the system's master file directory (MFD) is searched. The MFD is indexed by user name or account number, and each entry points to the UFD for that user.
- When a user refers to a particular file, only his own UFD is searched.
- Thus, different users may have files with the same name.
- The two – level directory structure solves the name-collision problem

Disadvantage:

- Users cannot create their own sub-directories.



Path Name

■ If a user can access another user's files, the concept of path name is needed.

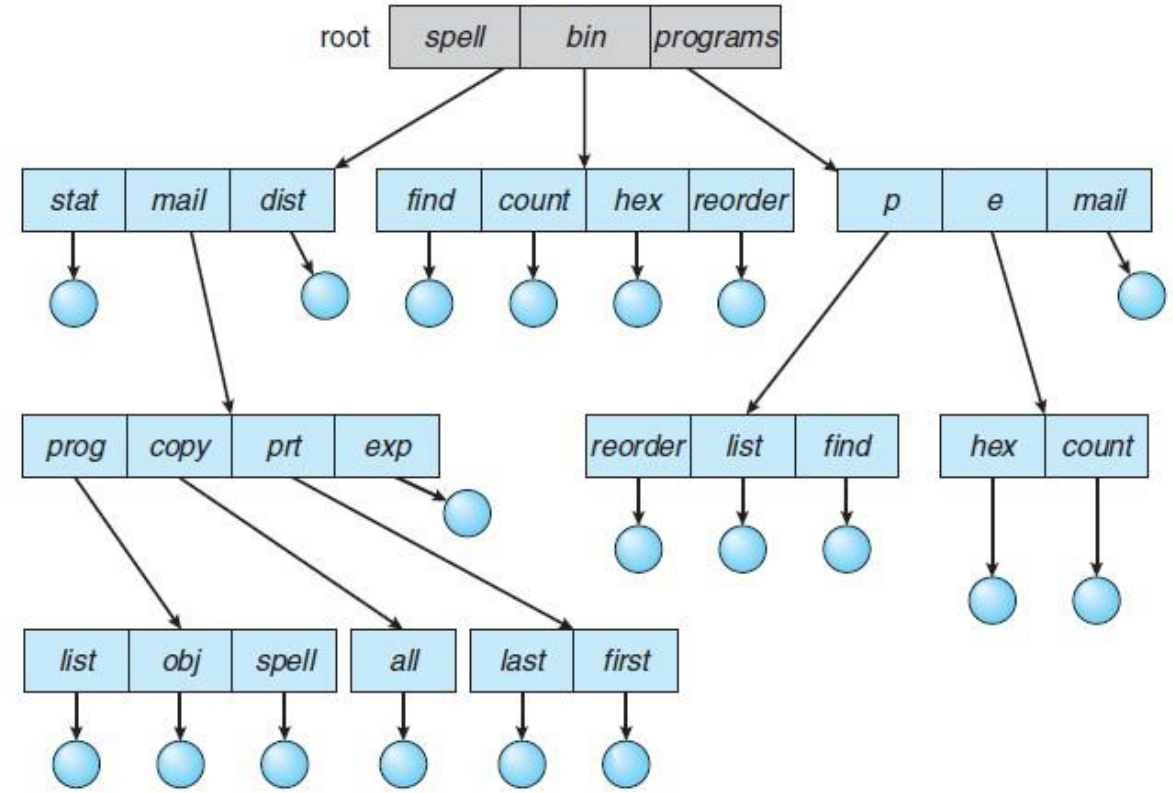
■ In two-level directory, this tree structure has MFD as root of path through UFD to user file name at leaf.

■ Path name :: username + filename

■ Standard syntax -- /user/file.ext

3. Tree – Structured Directory

- A tree is the most common directory structure.
- The tree has a root directory. 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.
- A directory (or sub directory) contains a set of files or sub directories.
- 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.
- Path names can be of two types: absolute path names or relative path names.
- 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.



4. Acyclic Graph Directory.

An acyclic graph is a graph with no cycles.

- To implement shared files and subdirectories this directory structure is used.
- An acyclic – graph directory structure is more flexible than is a simple tree structure, but it is also more complex.

Implementations of shared files or directories

1. Links

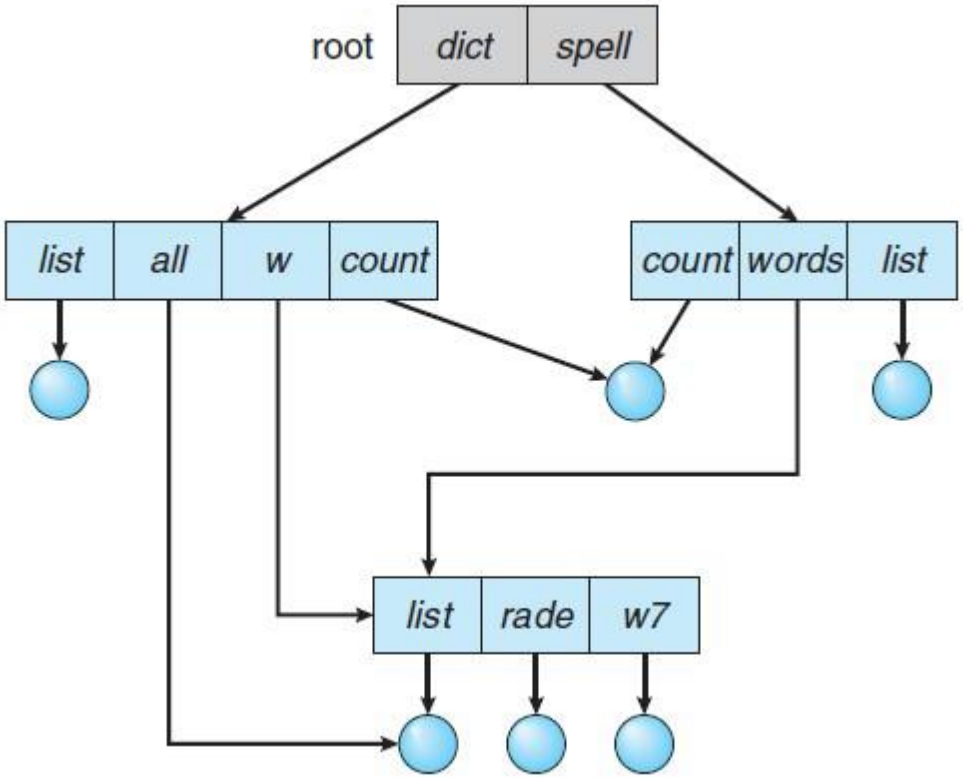
- A new type of directory entry is created. It is effectively a pointer to another file or subdirectory
- Links are implemented as an absolute or relative path name.
- The deletion of a link does not need to affect the original file; only the link is removed.

2. Duplicate all information in sharing directories

Problems to consider with link implementation:

- Upon traversal of file system, do not want to traverse shared structures more than once

- On deletion



Deletion:

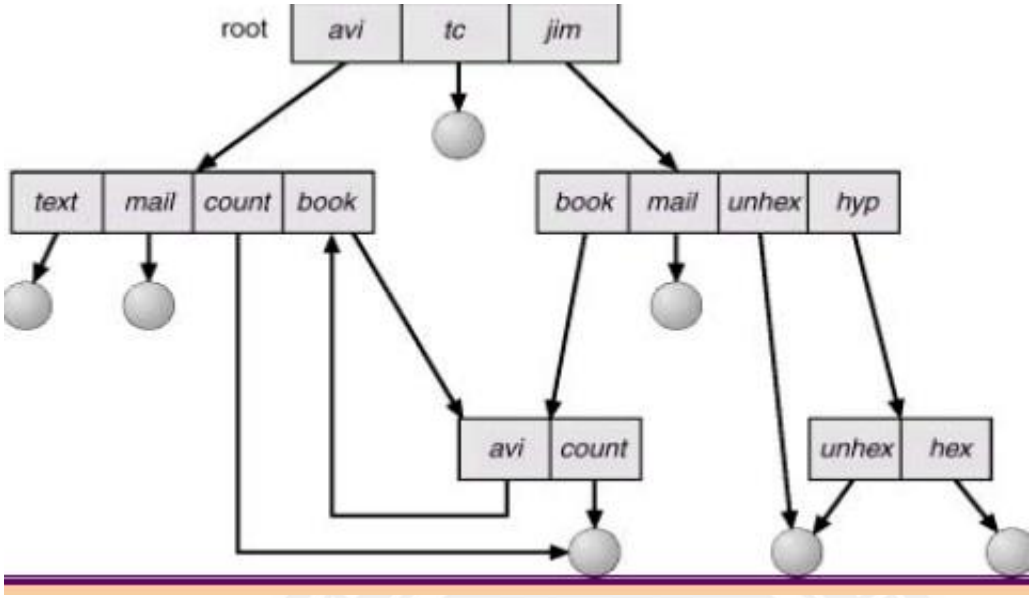
- The deletion of a link does not need to affect the original file; only the link is removed.
- Another approach to deletion is to preserve the file until all references to it are deleted. keep a count of the number of references.

  When count=0, file is deleted.
- Maintain a file reference list containing one entry for each reference to the file. On deletion just delete the entry from file reference list. File is deleted when this list becomes empty.

With a shared file only one actual file exists, so any changes made by one person are immediately visible to the other.

5. General Graph Directory

- When links are added to an existing tree-structured directory, a general graph structure can be created.



- A general graph can have cycles and cycles cause problems when searching or traversing file system.
- How do we guarantee no cycles?
    - ✦ Allow only links to files not subdirectories.
    - ✦ Use Garbage collection. {computationally expensive}
    - ✦ Every time a new link is added, use a cycle detection algorithm to determine whether a cycle now exists.

Disadvantage: Computationally expensive

An alternative approach is to bypass links during directory traversal.