Logical structure of a directory

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
- Limitations
  - when the number of files increases – difficult to keep track of files
  - when the system has more than one user and both users use the same name – Unique file name rule is violated

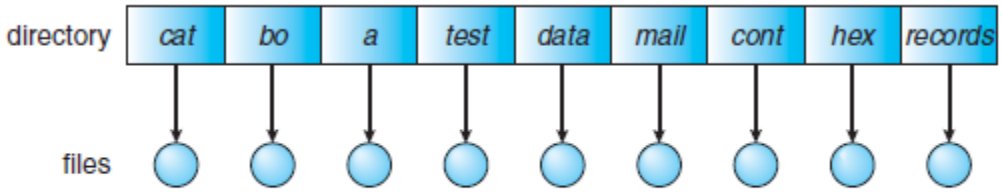

Figure 11.9 Single-level directory.

Two – Level Directory

- Create a separate directory for each user.
- Master File Directory (MFD) and User File Directory (UFD)
- Each user has his own UFD. Each UFD – Contains only the files of a single user.
- The MFD is indexed by user name or account number, and each entry points to the UFD for that user.
- For a new user the OS creates a new UFD and adds an entry for it to the MFD.
- 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.
- Thus, different users may have files with the same name, as long as all the file names within each UFD are unique.
- When a user searches for a particular file, only his own UFD is searched.

- 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 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. The files are the leaves of the tree.

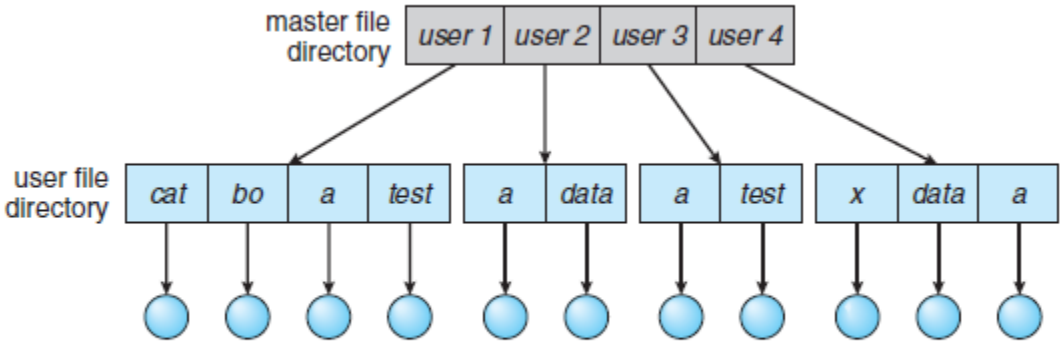

Figure 11.10 Two-level directory structure.

Tree – Structured Directories

- Tree structured directories extend two – level directory structure to a tree of arbitrary height.
- This generalization 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).

- Special system calls are used to create and delete directories and move from current directory to new directories (Unix cd command).
- To delete a directory, have to first delete all subdirectories and files in that directory. Only empty directory can be deleted.
- open() system calls search the current directory for the specified file.

- Path names of files – two types
  - absolute path name – define path from the root to the specific file.
  - relative path name – define path from current directory.

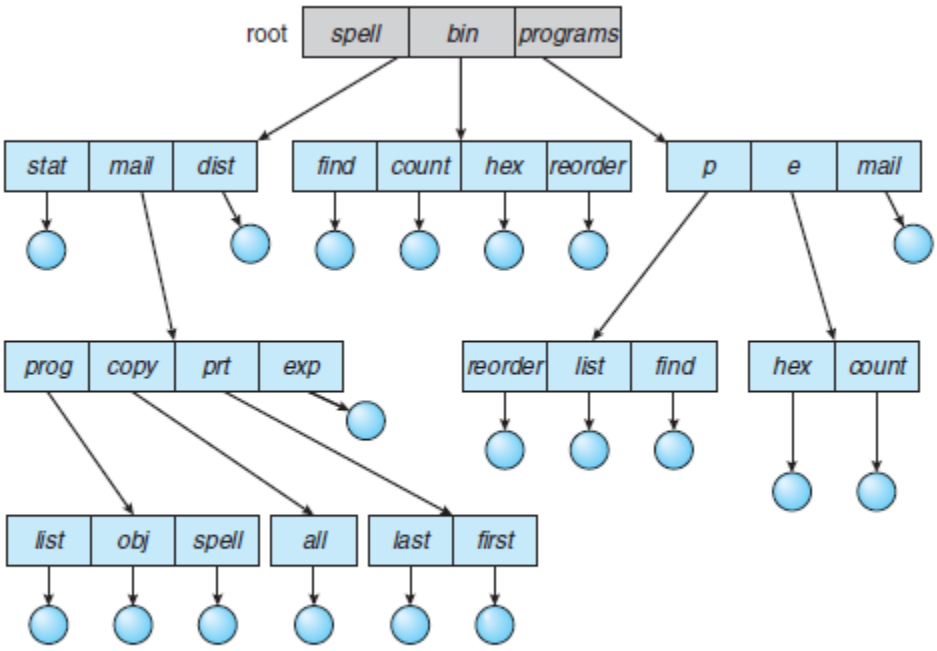

Figure 11.11 Tree-structured directory structure.

Acyclic – Graph Directories

- Used in situations when two or more users wants to create and access a shared subfolder in their own directories.
- A shared directory or file exists in the file system in two (or more) places at once.
- 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.

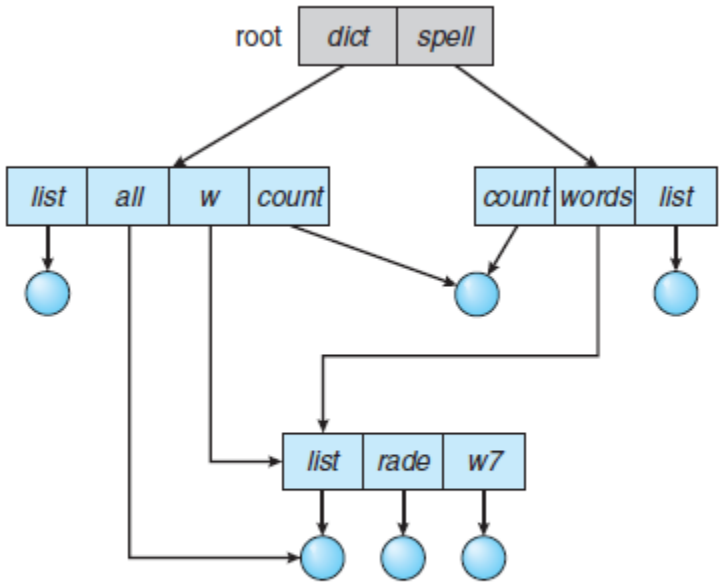

Figure 11.12 Acyclic-graph directory structure.

- With a shared file, only one actual file exists, so any changes made by one person are immediately visible to the other. Sharing is particularly important for subdirectories; a new file created by one person will automatically appear in all the shared subdirectories.
- When people are working as a team, all the files they want to share can be put into one directory. The UFD of each team member will contain this directory of shared files as a subdirectory.

- Shared files and subdirectories can be implemented in several ways.

- Unix – creates a new directory entry called a link. A link is effectively a pointer to another file or subdirectory.
- Another common approach is to duplicate all information in both sharing directories. Problem – maintaining consistency when a file is modified.
- An acyclic – graph directory structure is more flexible than a simple tree structure, but it is also more complex.
- Problems
    - Graph traversal – finding a file –
    - Deletion – when and how the space allocated to a file can be deallocated and reused (dangling pointers)
        - if link points to a disk address – File is deleted – link points to nonexistent file – If space is reallocated to another file, link points to new files
    - solution –
        - Search for these links and remove them as well, but unless a list of the associated links is kept with each file, this search can be expensive.
        - Leave the links until an attempt is made to use them. If file does not exist the access is treated just as with any other illegal file name.
        - Keep count of number of symbolic links to file. To delete a file if count is non-zero delete the link only. If zero delete the file.

General Graph Directory

A serious problem with using an acyclic – graph structure is ensuring that there are no cycles. If we start with a two – level directory and allow users to create subdirectories, a tree – structured directory results. It should be fairly easy to see that simply adding new files and subdirectories to an existing tree – structured directory preserves the tree – structured nature. However, when we add links, the tree structure is destroyed, resulting in a simple graph structure

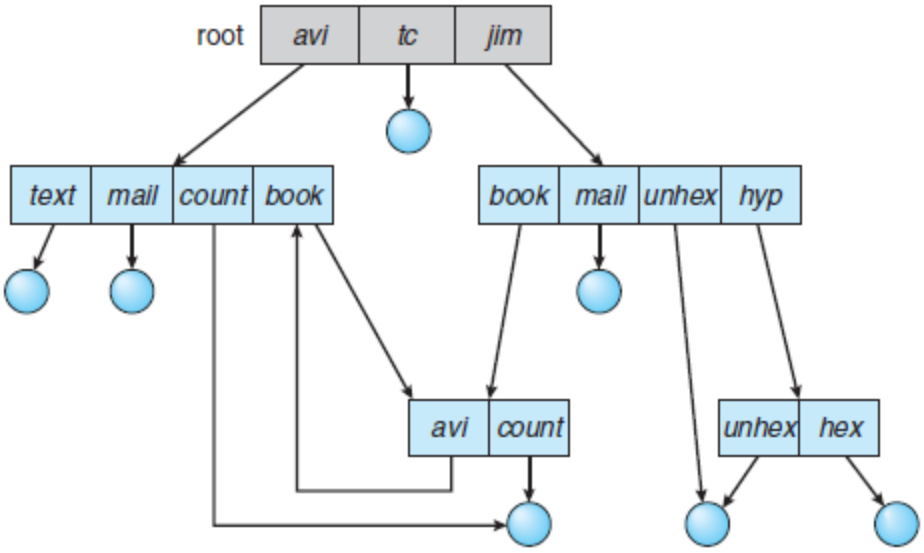

Figure 11.13 General graph directory.

- advantage of an acyclic graph is the relative simplicity of the algorithms to traverse the graph and to determine when there are no more references to a file.
- Problems –
    - How to avoid traversing / searching shared sections of an acyclic graph more than once.
    - How to delete a directory with multiple links / references