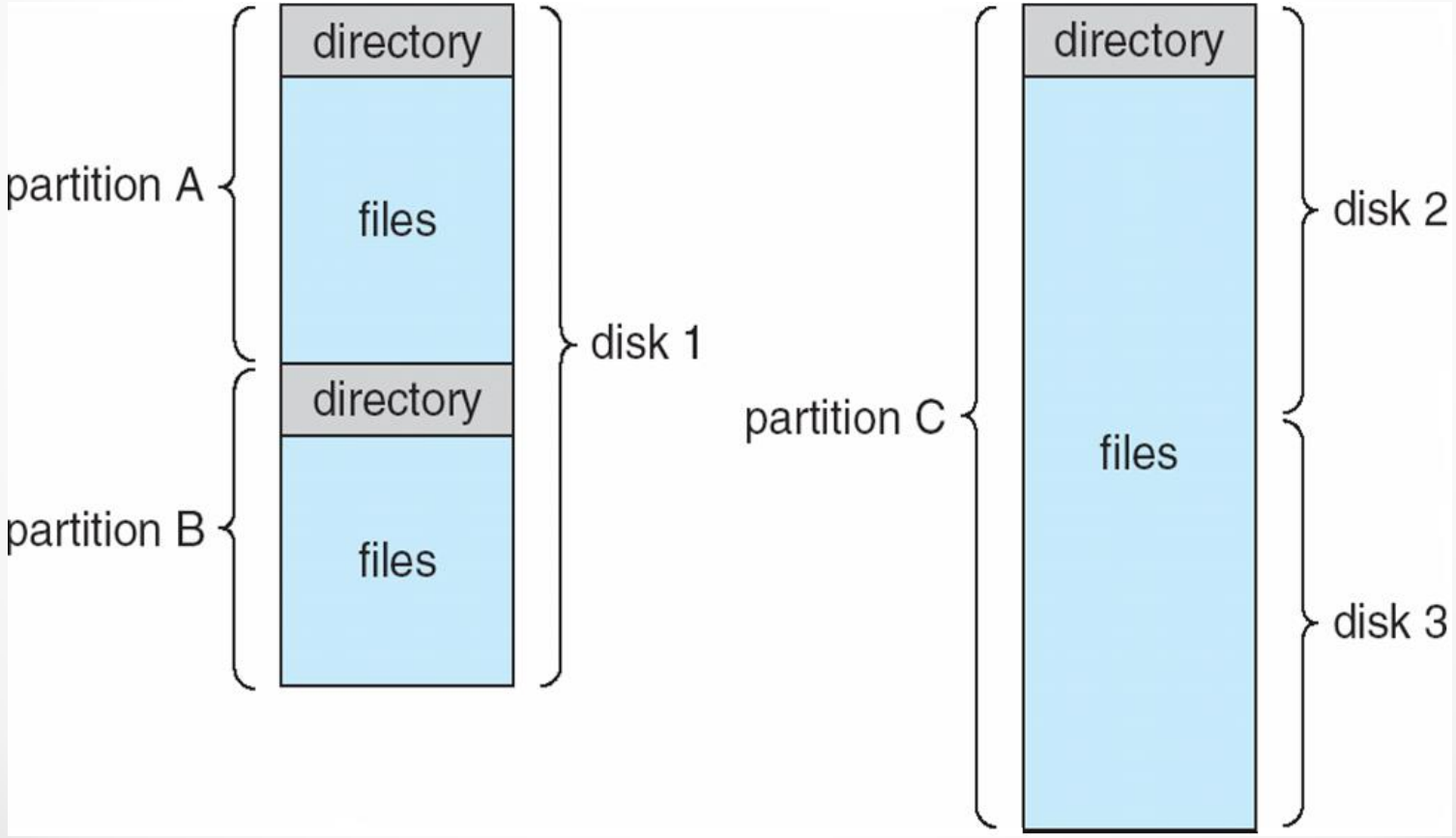
File System Interface

It consists of

- collection of files, for storing user data
- directory structure, how files are organized
- partition(possibly), separation of physical and logical collection of directories.

File System Organization

A directory can viewed as a "*symbol table*" that translates file names into their directory entries.



Directory Structure

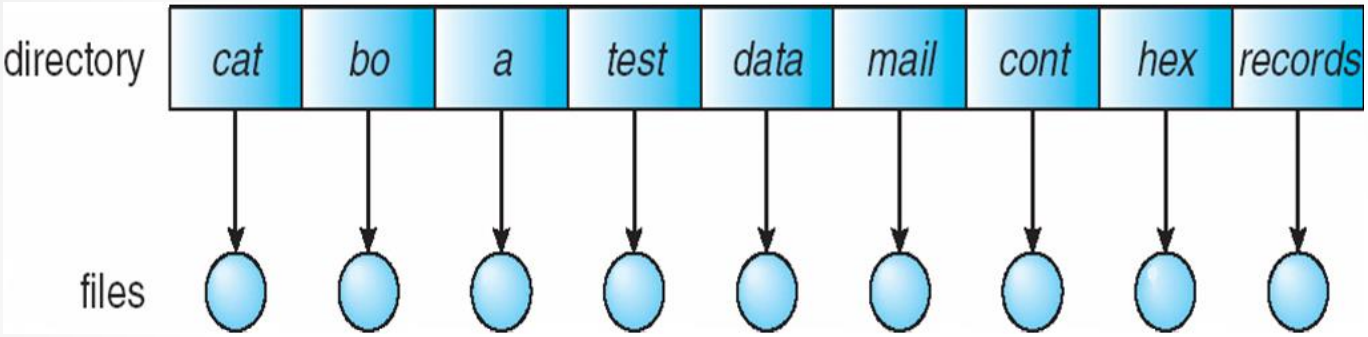
• Each disk on the system contain at least one partition, which is low-level structure of OS.

•Partitions (or Volumes) are the abstraction of virtual disks.

•Partitions can store multiple operating systems such that a system can boot more than one OS.

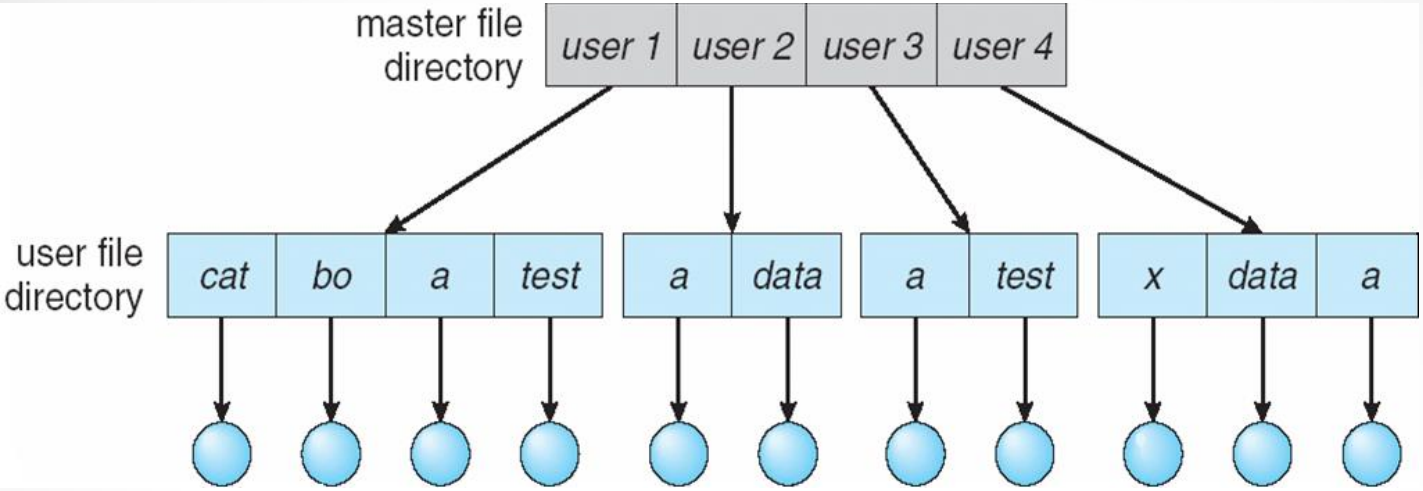
•A **VTOC** (Volume Table of Contents) contains information of all files in that partition.

Single-Level Directory



- All files are contained in same directory, means a single directory for all users.
- Simple to implement and searching is faster as size is small.
- *Name collision* because two files can not have the same name.
- No natural system for keeping track of file names i.e. the *grouping problem*.

Two-Level Directory



• Master File Directory (MFD) has pointers to individual User File Directories (UFD's).

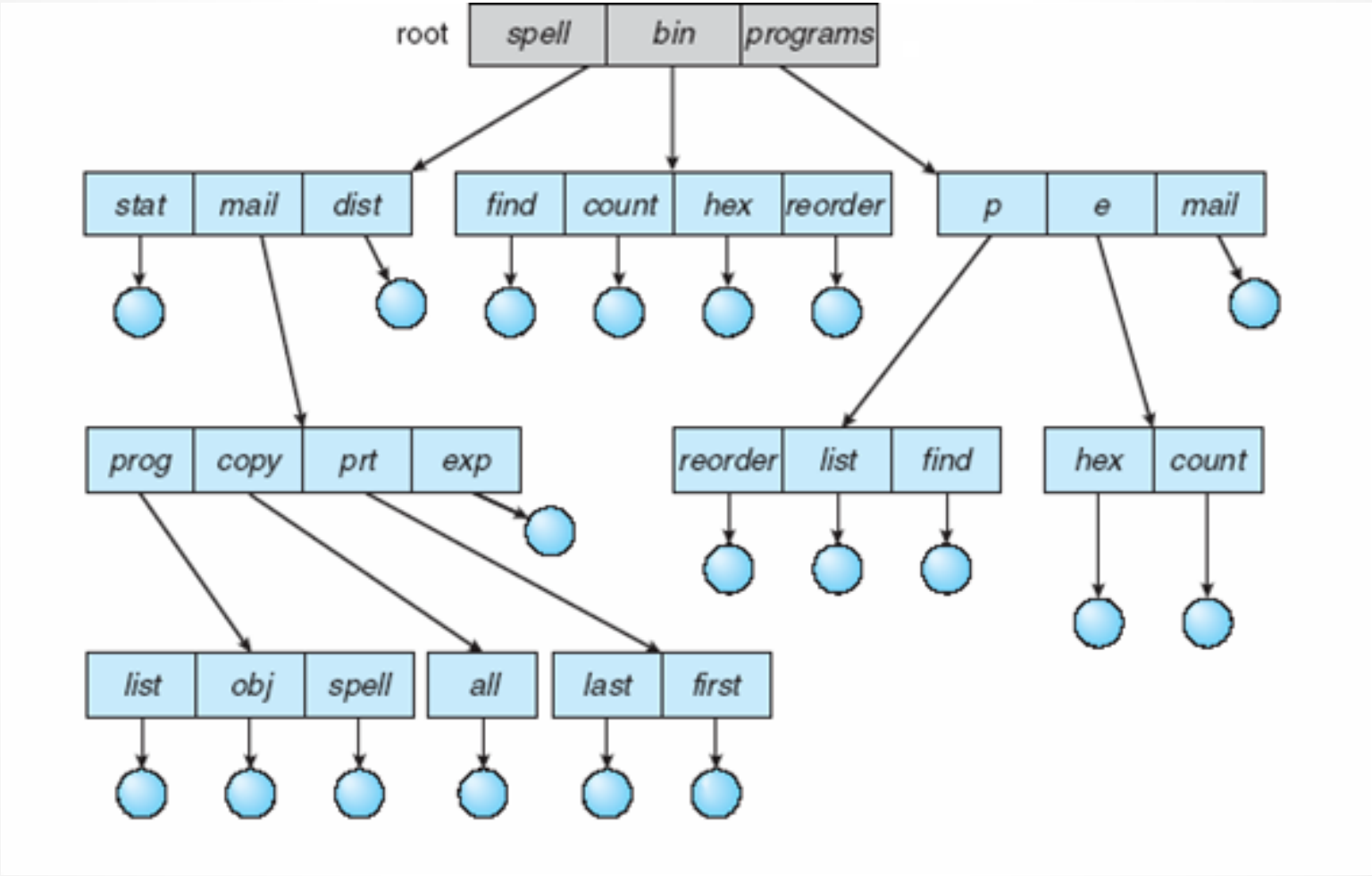
• Each user has his own separate directory.

•The MFD is indexed by username or account number, and each entry points to the UFD for that user.

Two-Level Directory

- Solves the problem of name collision.
- Isolates users from one another provides a sort of protection.
- Searching of files become more easy due to path name like /User-name/directory-name/.
- Restricts user cooperation because of isolation of users.

Tree-structured directory



Tree-structured directory

•It is a generalization of two level directory structure with arbitrary height allowing users to create number of sub-directories and files.

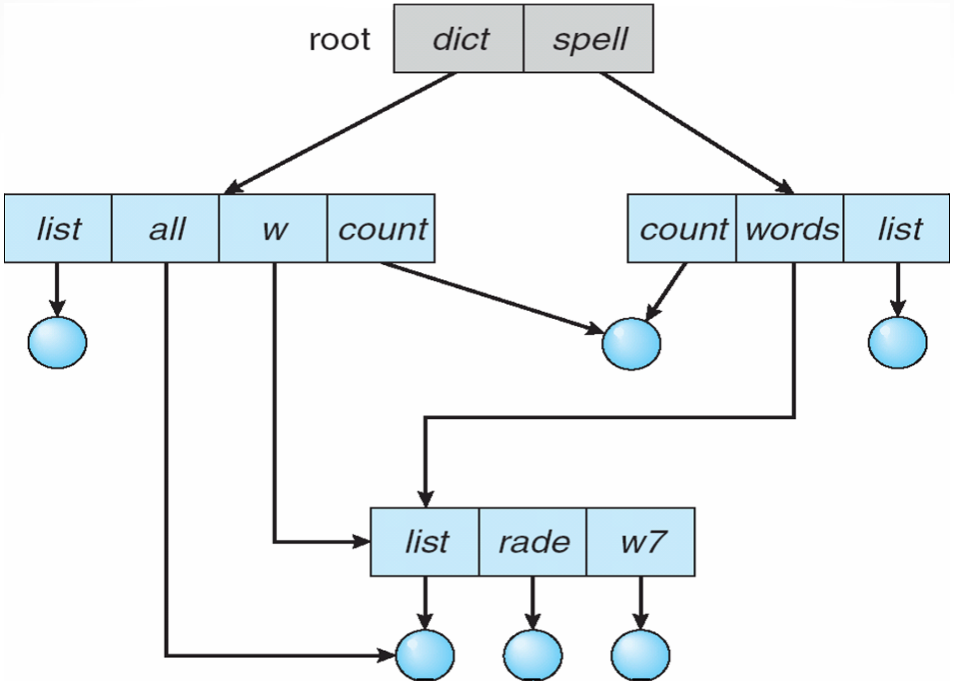
•One bit in directory entry defines entry as file or directory. 0 for file and 1 for subdirectory.

•The tree has a root directory, and every file in the system have a unique path.

Tree-structured directory

- It has two type of path
  - *Absolute path*, starts from the root and mentioning name of all directories till the specified file.
  - *Relative path,* defines the path from the current directory or working directory.
- Searching is efficient.
- *Allows accessing* of files of other groups.

Acyclic-Graph Directory



- A tree structure prohibits the sharing of files or directories.

- Acyclic graphs allow directories to have shared subdirectories and files.

Acyclic-Graph Directory

•Implementations of shared files or directories are done through ***links***.

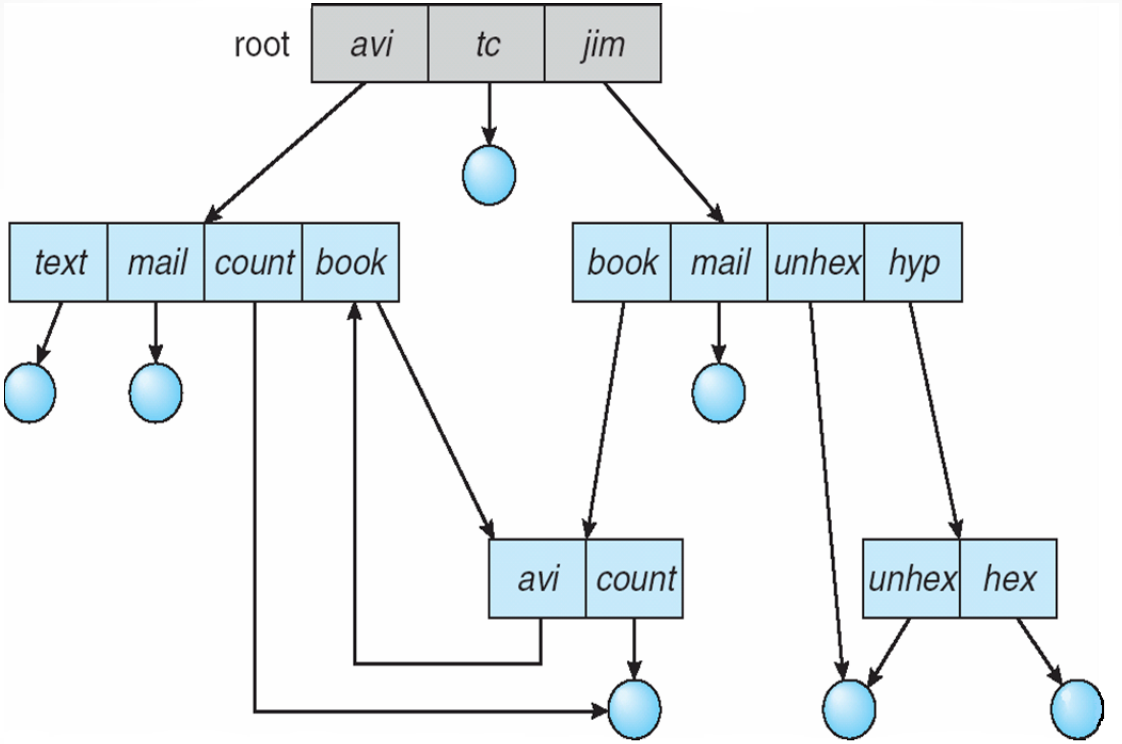
•*Link* is effectively a *pointer* to another file or subdirectory, which is implemented by containing an *absolute or relative path name*.

•A file may have multiple absolute path names leading to the *aliasing problem*.

•If the link is ***softlink*** then after deleting the file we left with a *dangling pointer* to non-existent file.

•In case of *hardlink*, to delete a file we have to delete all the reference associated with it.

General Graph Directory



Cycles are allowed within a directory structure where multiple directories can be derived from more than one parent directory.

General Graph Directory

- Provides more flexibility than others because of the cycles to search a specific directory or file.

- Also cause for more cost.

- Poorly designed algorithm might result in an infinite loop for searching.

- ***Garbage collection*** is required to reclaim the allocated space as self-referencing(cycle) mislead that the file exist even after the deletion.

****

for more interpretation follow the link below
https://www.youtube.com/watch?v=ZlDqiWK57Zw&t=5s