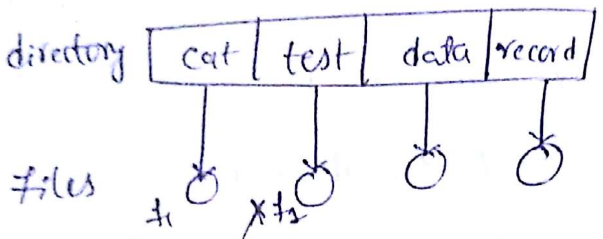
① Single level Directory

It is simplest directory structure all the files are contained in the same directory.

Since all the files are in the same directory, they must have unique names.



Drawback:- If there are lots of lots of files all files should have unique name. Maintaingng it is very difficult.

② Two Level Directory

Create a seperate directory for each user.

In this structure, each user has his own user file directory (UFD). when a user job starts ~~for~~ 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 files may have same name, as long as all the file names within each UFD are unique.

Create:- To create a file for a user, the OS searches only that user's UFD to know whether another file of same name exists.

Delete: To delete a file, the OS confines its search to the local UFD thus, it can not delete another user's file that has the same name

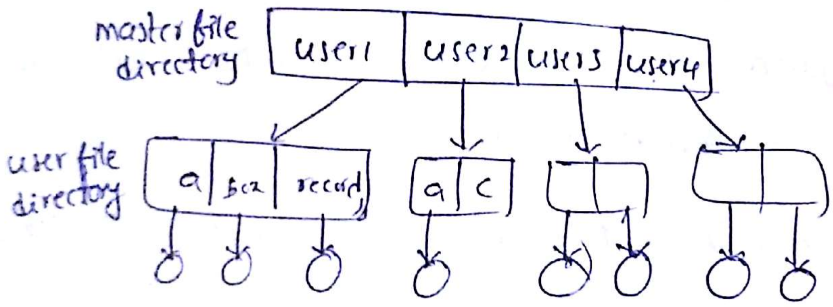
Disadvantages:-

Although it solves the name-collision problem, it still has disadvantages.

→ Isolation is an advantage when the users are completly independent but is a disadvantage when the users want to cooperate on some task & to access one another's files.

Some systems simply do not allow local user files to accessed by other users.

Two-level structure



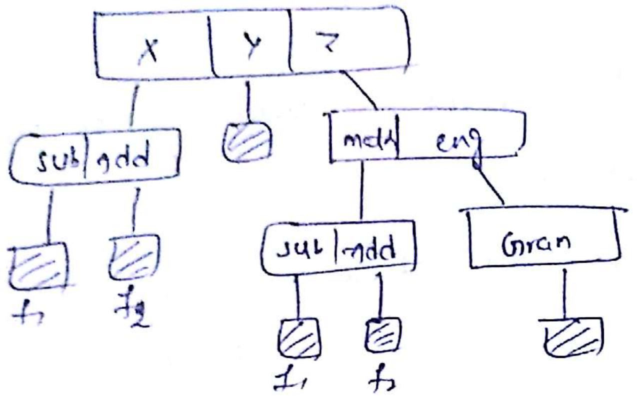
To name, a file uniquely a user must know the path name of file desired

③ Tree-structured Directory

Allows users to create their own sub-directories and organize their files accordingly. A Tree is most common directory structure.

→ A Directory contains a set of files or subdirectories. 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



Deletion: If a directory is empty, its entry in the directory that contains it can simply be deleted. suppose the directory to be deleted is not empty but contains several files / sub directories.

somes system does not allow (MS-DOS) deletion of directory unless it is empty

⇒ This allows accessing files of others by specifiing path.

Disadvantages: ~~If~~ sharing is not possible. (files/directories)

④ Acyclic-Graph Directories

consider 2 programmers who are working on a joint project. The files associated with that project can be stored in a subdirectory, seperating them from other projects & files of the 2 programmers both are equal responsible, both want subdirectory to be in their own directories.

The common subdirectory should be shared. A shared directory or file will exist in the file system in 2 (or more) places at once.

→ An acyclic graph that is, a graph with no cycles - allows directories to share subdirectories & files

→ shared file is not same as two copies of the file

change made by other visible to all

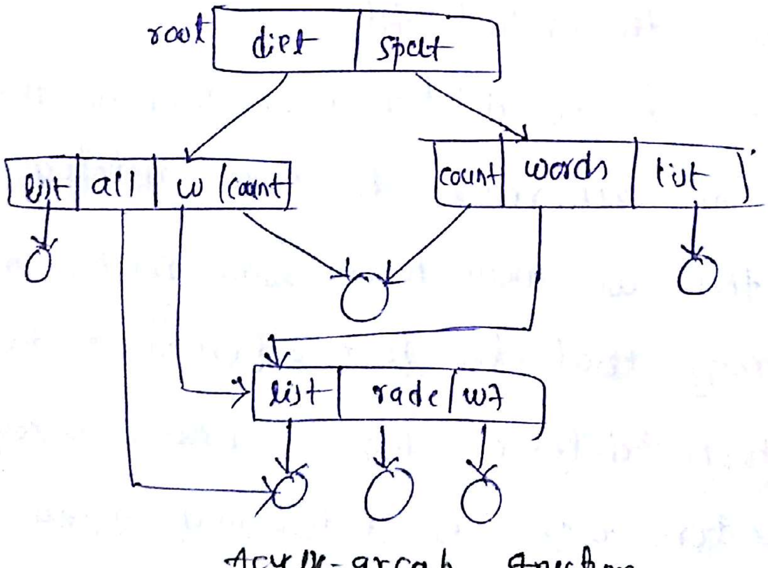
changes on one file does not effect other file

shared files & subdirectories can be implemented in several ways.

① Create a new directory entry called a "link"

⇒ Links are easily identified by their format in the directory entry.

→ The OS ignores these links when traversing directory trees to preserve the acyclic structure of the system



Acyclic-graph structure

② Another common approach to implementing shared files is simply to "duplicate" all information about them in both sharing directories.

-> A major problem with duplicate directory entries is maintaining consistency when a file is modified.

* A file now have multiple absolute path names. consequently, distinct file names may refer to the same file. This situation is similar to the aliasing problem for programming languages.

Another problem is deletion

One possibility is to remove the file whenever anyone deletes it, but this action may leave dangling pointers to the now-non existent file.

=> sharing is implemented by symbolic links, this is somewhat easier to handle - the deletion of a link need not affect the orginal file.

-> Another apporach to deletion is to preserve the file until all references to it are deleted. To implement this, we must have some mechanism for determining that the last reference to the file has been deleted. when a link or a copy of the directory entry is established, a new entry is added to the file-reference list. if link is deleted remove from the list.

The file is deleted when its file-reference list is empty.

The trouble with this approach is the variable & potentially large size of the file-reference list.

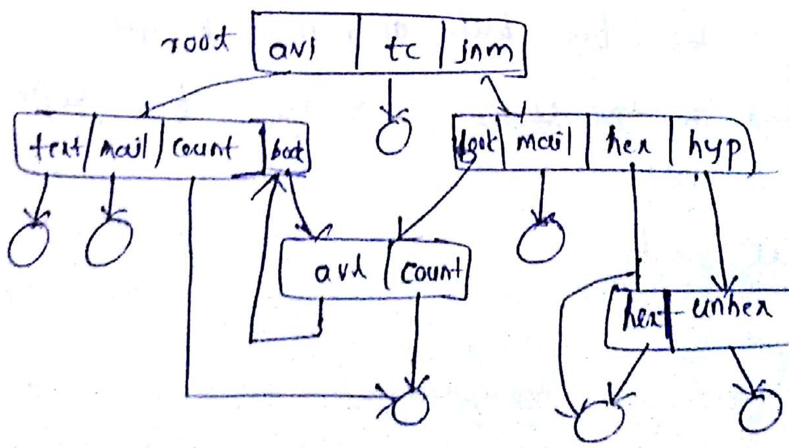
Disadvantages some system do not allow shared directories or links.

ex: MS-DOS - is tree structure

⑤ General graph Directory

A similar problem exists when we are trying to determine when a file can be deleted. with acyclic-graph directory, a value of '0' in the reference count means that there are no more references to the file or directory, & the file is deleted. However, when cycles exist, the reference count may not be '0' even when it is no longer possible to refer to file / directory. possiblity of self-referencing (a cycle)

In this case, we generally need to use a "grabage-collection scheme" to determine when the last reference has been deleted & the disk space can be reallocated.



root
avi
tc
jim
text
mail
count
book
avi
count
book
mail
hex
hyp
hex
unhex