UNIT-IV

Storage Management - file system- concept of a file, system calls for file operations - open(), read(), write(), close(), seek() unlink(), Access methods - Directory and Disk structure, file system Mounting, file sharing, Protection

File system implementation - File System Structure, file system implementation, Directory file system implementation, Allocation methods, free-space management, efficiency and performence mass storage structure - overview of mass storage structure, Disk structure, Disk attachment, Disk scheduling, Disk management, swap space management

File System:

→ Computers can store information on various storage media, such as magnetic disks, magnetic tapes, and optical disks. so that the computer system will be convenient to use.

→ File is a collection of related information that is recorded on secondary storage. or file is a collection of logically related entities. from user's perspective a file is the smallest allotment of logical secondary storage.

→ A file is a sequence of bits, bytes, lines or records.

→ different types of information stored in a file. source programs, object programs, executable programs, numeric data, text, payroll records, graphic images, sound recordings, and so on.

File Attributes:

Name: The symbolic file name is the only information kept in human readable form.

identifier: identifies the file within the file system, it is the non-human-readable name for the file.

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

Location: This information is pointer to a device and to the location of the file on that device.

Size: The current size of the file and possibly the maximum allowed size are included in this attribute.

protection: It determines who can do reading, writing, executing and soon

Time, date, and user identification: This information may be kept for Creation, last modification, and last use. These data can be useful for protection, security, and usage monitoring.

File operations:

File is an abstract data type. operating system can provide basic six operations.

- Creating a file:
- Writing a file
- Reading a file
- Repositioning within a file
- Deleting a file.
- Truncating a file

System calls for file operations:

Basically there are total 5 types of I/O system calls

1. create()
2. open()
3. close()
4. read()
5. write()

Create(): used to create a new empty file.

indicate permission of new file

↑

→ it return -1 when an error occur otherwise it return first unused file descriptor.

open(): used to open the file for reading, writing or both.

syntax:
int open(const char* path, int flags [, int mode]);

where path - path to file which you want to use, if absolute path begin with '/' [same dir of file], use relative path i.e only file name with extension [same dir of file].

flags - O_RDONLY - read only
O_WRONLY - write only
O_RDWR - read and write
O_CREAT - Create file if doesn't exist

close(): To close file which pointed by file descriptor

syntax: int close(int fd); ← file descriptor

it return - 0 - on success
it return - '-1' - on error.

read(): From the file indicated by the file descriptor fd, the read() function reads count bytes of input into the memory area indicated by buffer.

syntax: Size_t read(int fd, void* buf, size_t cnt); ← length of buffer
↑ buffer to read data

Returns - how many bytes were actually read
return - 0 on reaching end of file
return - '+1' on error or signal interrupt
returns

write():

Syntax: size_t write (int fd, void * buf, size_t cnt);

fd: file descriptor
buf: buffer to write data
cnt: length of buffer

Returns - how many bytes were actually written.
Return - 0 on reaching end of file
return - 1 on error or signal interrupt

lseek(): lseek is a system call that is used to change the location of the read/write pointer of a file descriptor. The location can be set either in absolute or relative terms.

Syntax: lseek (int fieldes, off_t offset, int whence);

the fd of the pointer that is going to be moved
The offset of the pointer
The method in which offset is interpreted

it returns the offset of the pointer from the beginning of the file.

unlink(): deletes a name from the filesystem. if that name was last link to a file and no processes have the file open the file is deleted and the space it was using is made available for reuse.

It return zero on success, otherwise -1.

Syntax: #include <unistd.h>
int unlink (const char * pathname);

Note: link() - Creates a new link to an existing file.

Access Methods:

When file is used, information is read and entered into computer memory and there are several ways to access this information of the file. Some system provide only one access method for files. other system support many access methods and choosing the right one for a particular application is a major design problem.

There are three way to access a file into computer system

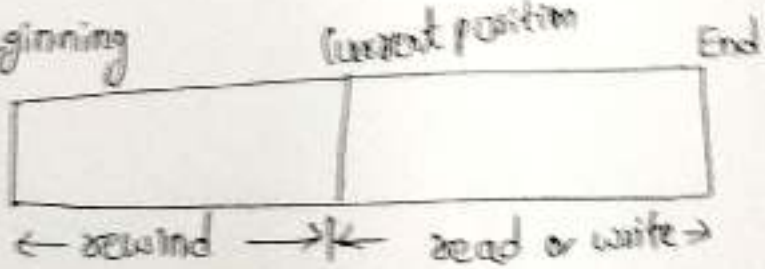
- Sequential Access
- Direct Access
- Index Sequential Access

Sequential Access:

→ It is Simplest access method. Data is accessed one record right after another record in an order. For example, editor and compiler usually access the file one after another.

→ When we use read command, it move ahead pointer by one using readnext.

→ When we use write command, it will allocate memory and move the pointer to the end of the file by using writenext



→ The Disadvantage it provide poor performences.

Direct Access(or) relative access:

→ In DA, a file is made up of fixed. length logical records that allow program to read and write records in no particular order

→ It based on the disk model of a file since disk allows random access to any file block.

→ In DA, the file is viewed as numbered sequence of block or record. thus, we may read block 14 then block 59 and then we can write block 17. There is no restriction on the order of reading and writing for direct access file.

→ A block number provided by the user to the operating system is normally a relative block number. the 1st relative block number is 0 and then 1 and so on.

Index Sequential Access:

→ It is built on top of sequential access.

→ it control the pointer by using index.

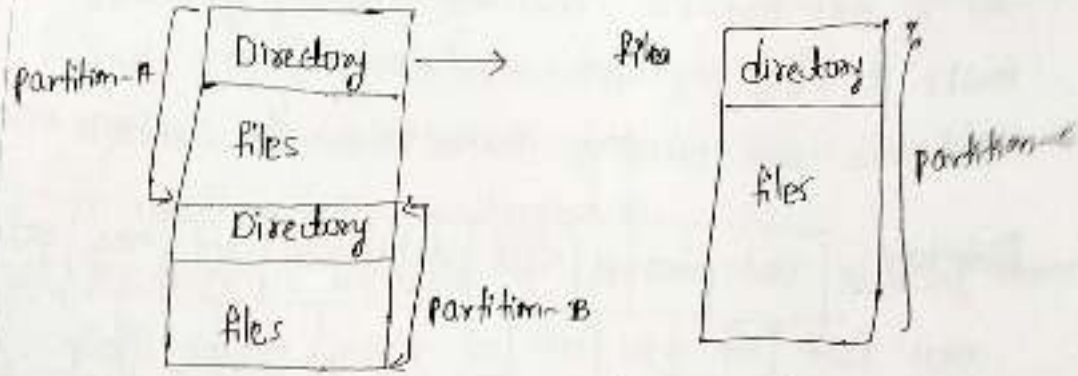
→ To find a record in the file, we first search the index and then by the help of pointer we access the file directly.

Directory and disk structure:

Directory can be defined as the listing of the related files on the disk. The directory may store some or the entire file attributes.

To get the benfit of different file systems on the different operating systems, A hard disk can be divided into the number of partitions of different sizes. The partitions are also called volumes or mini disks.

Each partition must have at least one directory in which all the files of the partition can be listed.



file system organization

Every directory supports a number of common operations on the file.

- file creation
- search for a file
- Delete a file
- List a Directory
- Rename a file
- Traverse the file system

There are several logical structures of directory, these are given below

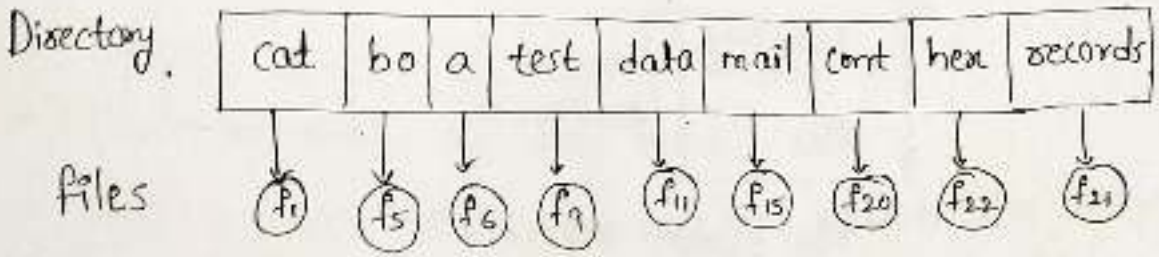
1. single level directory
2. Two level directory
3. Tree structured directory
4. Acyclic graph directory
5. General graph directory

Single - level directory:

→ It is a simplest directory structure.

→ In it all files are contained in same directory which make it easy to support and understand.

→ Each file and directory must have the unique name.



Single - level directory

Advantages:

→ implementation is very easy because of single level

→ if files are smaller in size, searching will faster

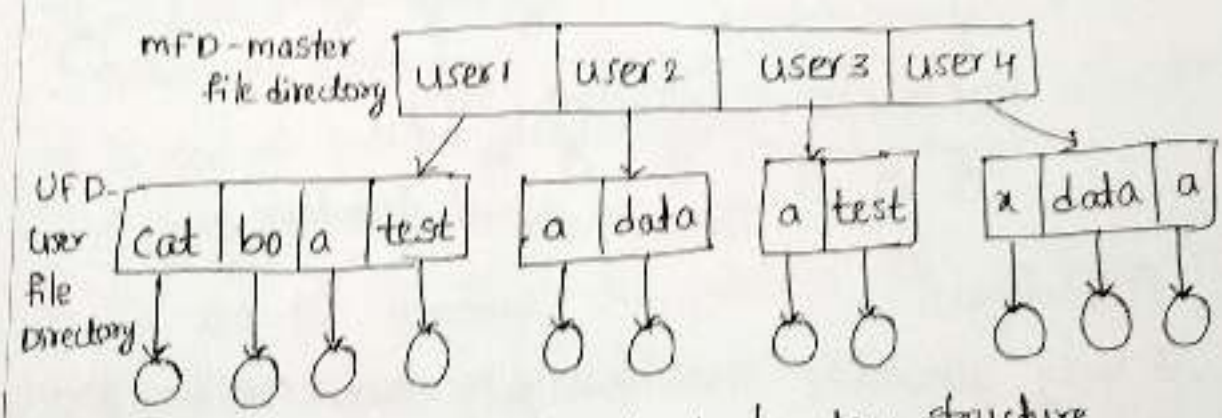
→ The file operations like file creation, searching, deletion, updating are very easy in single - level.

Disadvantages:

- If two files having same name collision will occur
- Searching will become time taking if directory will large
- We can not group same type of files.

Two-level Directory:

- In single-level directory, it leads to confusion of files names among different users the solution to this problem is to create a separate directory for each user.
- In two-level, each user has there own user file directory (UFD). The UFD's has similar structures, but each list only the files of single user.
- System's master files directory (MFD) is searches whenever a new user id = s logged in.
- The MFD is indexed by username or account number, and each entry points to the UFD for that user.



Two-level directory structure

advantages: → we can give full path like username/dir/file

→ Different users can have same directory as well as filenam

→ searching of files is very easy because of path.

Disadvantage:

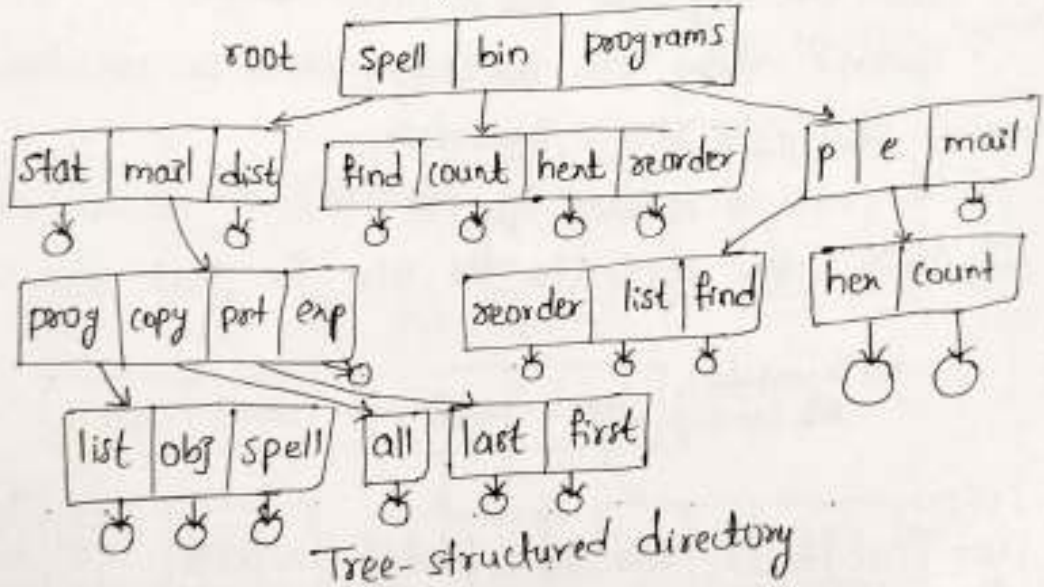
→ A user is not allowed to share files with other users.

→ Still it not very scalable, two files of the same type cannot be grouped together in the same user.

Tree- structured directory:

→ A tree structure is the most common directory structure The tree has a root directory, and every file in the system have a unique path.

→ The natural generalization is to extend the directory structure to a tree of arbitary height. this generalization allows the user to create there own subdirectories and to organize on their files accordingly.



Tree- structured directory

Advantages:

→ very generalize, since full path name can be given

→ very scalable, the probability of name collision is less

→ Searching becomes very easy, we can use both absolute path as well as relative.

Disadvantages:

→ We cannot share files.
→ It is inefficient, because accessing a file may go under multiple directories.
→ Every file does not fit into the hierarchical model, files may be saved into multiple directories.

Acyclic graph directory:

→ It is a graph with no cycle and allows to share subdirectories and files the same file or sub directory may in two different directories.
→ It is a natural generalization of the tree structured directory.
→ it is used in the situation like when two programmers are working on a joint project and they need to access files.

Advantages:
→ We can share files
→ Searching is easy due to different paths

Disadvantages:

→ We share the files via linking, in case of deleting it may create the problem.
→ In case of softlink, after deleting the file we left with a dangling pointer
→ In case of hardlink, to delete a file we have to delete all the reference associated with it.

General graph directory:

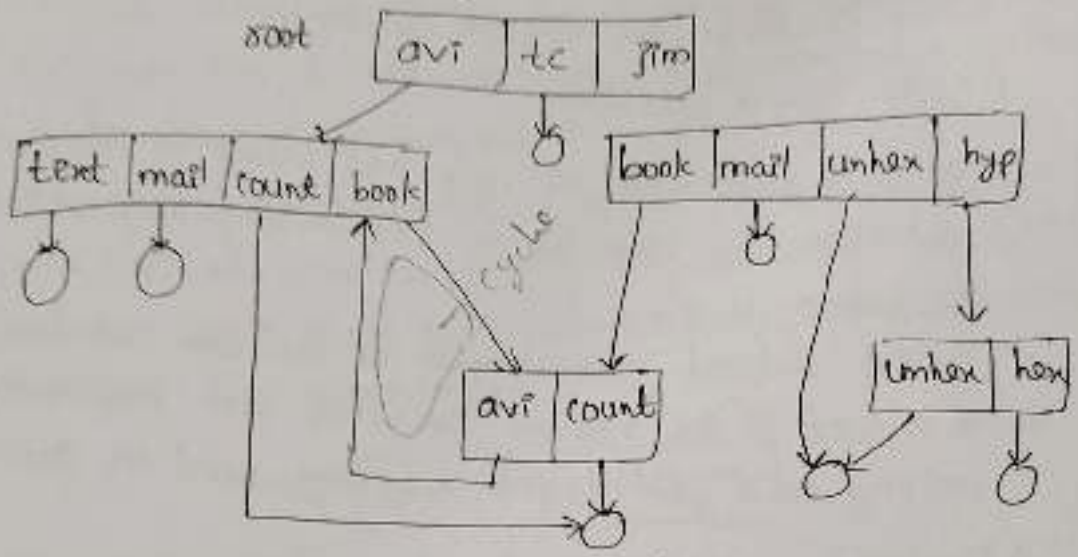
→ It allow the cycles within a directory structure where multiple directories can be derived from more than one parent directory.
→ To calculate the total size or space complex in this directory.

Advantages:

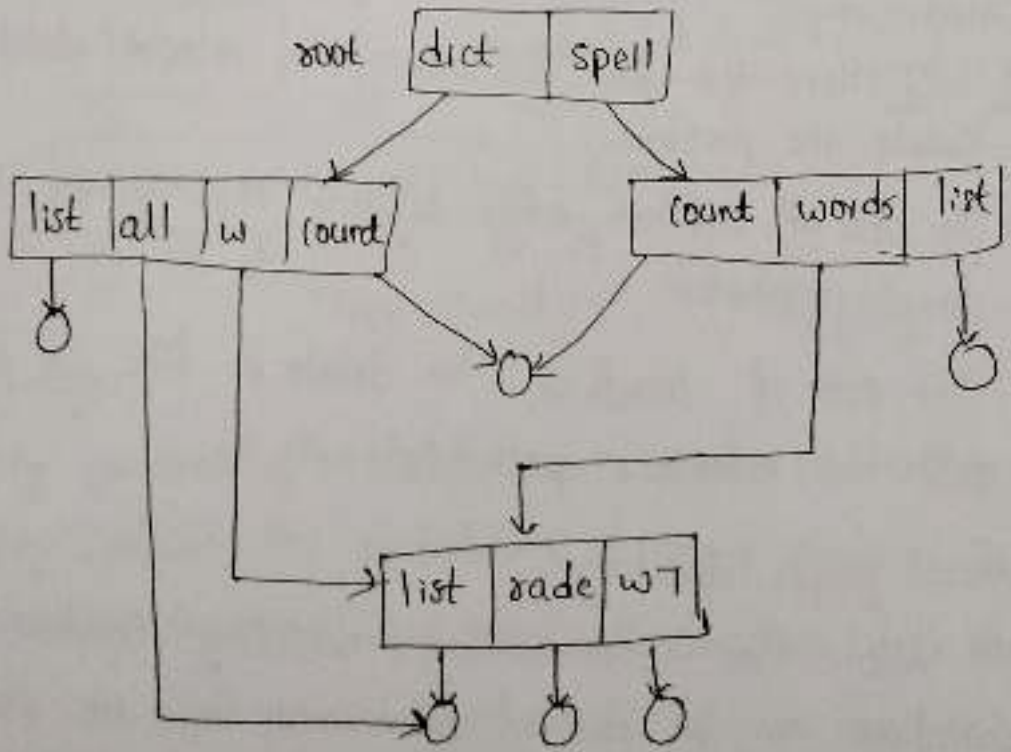
- It allows cycles
- it is more flexible than other directories structure.

Disadvantages:

- It is more costly than others
- it needs garbage collection.



General graph directory.



Acyclic-graph directory

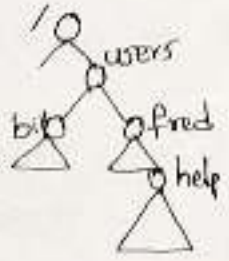
File system mounting:

→ Mounting is a process by which the operating system makes files and directories on a storage device such as hard drive, CD-ROM, or network share that make availability for users to access through the computer's file system.

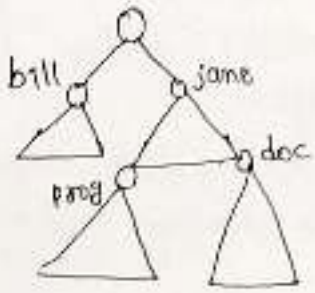
→ A file system must be mounted before it can be available to processes on the system.

→ The mount procedure is straight forward The operating system is given the name of the device and the mount point.

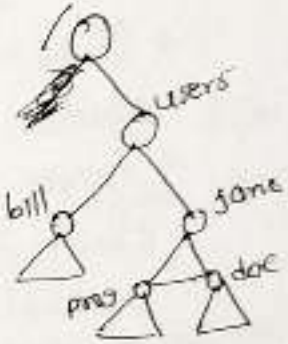
→ The operating system verifies that the device contains a valid file system.



Existing system



Unmounted volume



mount point

→ Windows operating systems automatically discover all device and mount all located file system at boot time.

→ In unix mount commands are explicit.

File sharing:

→ file sharing is very desirable for users who want to collaborate and to reduce the effort required to archieve a computing goal.

⇒ on distributed systems, files may be shared across a network

→ Network file system (NFS) is a common distributed file sharing method. sharing may be done through a protection scheme.

→ Multiple users, user IDs identify users, allowing permissions and protections to be per user. Group IDs allow users to be in groups, permitting group access rights.

→ In Remote file systems, Networking allows the sharing of resources spread across all the world the following method that are use to share file.

- manually via programs like FTP
- automatically, using distributed file systems
- Semi automatically via the world wide web.

→ client-server model allows clients to mount remote file systems from servers.

→ server can serve multiple clients. NFS is standard UNIX client-server file sharing protocal. CIFS is standard windows protocal.

→ Standard operating system file calls are translated into remote calls.

→ In failure modes, Local system can fail for a variety of reasons, including failure of the disk containing the file system, corruption of the directory structure or other disk management information.

→ Remote file systems have more failure modes, due to network failure, server failure.

→ Recovery from failure can involve state information about status of each remote request.