Chapter 2: Introduction to file system and file management

2.1 Introduction to file system

File

- A file is a named collection of related information that is recorded on secondary storage, usually as a sequence of bytes, with two views:
  - ✓ **Logical (programmer) view,** as the users see it (how they are used and what properties they have.).
  - ✓ **Physical (os) view,** as it actually resides on secondary storage.
- Data files may be numeric, alphanumeric, or binary. Files may be free from, such as text files, or may be formatted rigidly.
- In general, a file is a sequence of bits, bytes, lines, or records the meaning of which is defined by the file's creator and user.
- Many different types of information may be stored in a file- source programs, object programs, executable programs, numeric data, text, payroll records, graphic images, sound recording, and so on.
- A file has a certain defined structure, which depends on its type.
  - ✓ A **text file** is a sequence of characters of organized into lines (and possibly pages).
  - ✓ A **source file** is a sequence of subroutines and function, each of which is further organized as declarations followed by executable statements.
  - ✓ An **object file** is a sequence of bytes organized into blocks understandable by the system's linker.
  - ✓ An **executable file** is a series of codes section that the loader can bring into memory and execute.

Record

- A record is a collection of related fields treated as a unit.
- A file consists of sequence of records.
- In a text file a record corresponds to a line of text, in other cases a record has no physical basis, it is just a convenient collection of values chosen to suit the application.
- For example, employee records, student records.

File system

- A part of an operating system, which deals with files, is known as file system or file manager.
- It is the responsibility of a file system to manage all the files.
- It provides facilities to create & delete files, read & write contents of files, etc.
- The **file system** consists of two distinct parts: **a collection of files**, each storing related data, and a **directory structure**, which organizes and provides information about all the files in the system.

- It specifies conventions for naming files. These conventions include the maximum number of characters in a name which characters can be used and, in some system, how long the file name suffix can be.
- A file system also includes a format for specifying the path to a file through the structure of directories.

File Naming

- Files are used to store information on the disk. This information need to be read back later. User does not need to concern about details such as how and where the information is actually stored on disk, how disks work, etc.
- For this reasons, files are named. When a file is created, it is given a name. Later on, it is referred by this name. This is to provide convenience to human users.
- In reality, information is stored on physical devices such as disks. But, operating system hides (abstracts) physical device, and provides logical entity (files) to store information. Due to this reason, files are sometimes referred as logical devices.
- The exact rules for naming files vary from system to system.
- All current operating systems allow strings of one to eight letters, including digits and some special characters, as legal file names. Thus test, xyz123, chap_1 are possible file names. Many file systems support names as long as 255 characters.
- Some file systems support case sensitive file names, while others do not. Case sensitive means, It distinguishes between upper and lower case letters. UNIX comes under first category, while MS-DOS comes in second category. So, test1, Test1, TEST1 are three different files in UNIX. But, all these names refer to the same file in MS-DOS.
- Many operating systems support two-part file names, with the two parts separated by a period ('.'), as in test. c. The part following the period is called the file extension. It usually indicates something about a file.
- In MS-DOS, file names are 1 to 8 characters, plus an optional extension of 1 to 3 characters.
- In UNIX, the size of the extension depends upon the user and a file may contain two or more extensions, such as chap. c.Z.
- Some of the more common file extensions are given in following figure.

Extension	Meaning
.txt	General text file
.c	C source program
.bak	Backup file
.obj	Object file (Compiler output, but yet not linked)
.exe	Executable file
.gif	Graphical Interchange Format file
.hlp	Help file
.html	World Wide Web Hyper Text Markup Language file
.jpg	Still picture encoded with JPEG standard
.mp3	Music encoded in MPEG layer 3 audio format
.mpg	Movie encoded with the MPEG standard
.pdf	Portable Document Format file
.ps	PostScript file
.zip	Compressed archive file

File attribute

- When a file is named, it becomes independent of the process, the user, and even the system that created it. For instance, one user might create the file, and another user might edit that file by specifying its name.
- A file's attributes vary from one OS to another but typically consist of these:
  - ✓ **Name:** A file is named for the convenience of the user and is referred to by its name. The symbolic file name is the only information kept in human readable form.
  - ✓ **Identifier:** This unique tag, usually a number,identifies the file within the file system; it is the non-human –readable name for the file.
  - ✓ **Type:** Type information is needed for those systems that support different types. The types are depended on the extensions of the file. For ex, .exe for executable, .obj for object file.
  - ✓ **Location:** This information is a pointer to a device and to the location of the file on that device.
  - ✓ **Size:** The current size of the file (in bytes, words, or blocks), and possibly the maximum allowed size are included in this attribute.
  - ✓ **Protection:** Access-control information determines who can do reading, writing, executing, and so on.
  - ✓ **Usage Count:** This value indicates the number of processes that are currently using this file.

- ✓ **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.
- ✓ The information about all files is kept in the directory structure, which also resides on secondary storage; typically, a directory entry consists of the file's name and its unique identifier.

2.2 Operations on file

- A file is an abstract data type. To define a file properly, we need to consider the operations that can be performed on files. The basic file operations are given as below.
- The OS can provide system calls to create, write, read, reposition, delete, and truncate files.

Creating file:

A new file can be created by a system call embedded in a program or by an OS command issued by an interactive user.

Two steps are necessary to create a file.

1. Space in the file system must be found for the file.
2. An entry for the new file must be made in the directory.

Writing a file:

To write a file, we make a system call specifying both the name of the file and the information to be written to the file.The system must keep a write pointer to the location in the file where the next write is to take place. The write pointer must be updated whenever a write occurs.

Reading a file:

To read from a file, we use a system call that specifies the name of the file and where (in memory) the next block of the file should be put.

The system needs to keep a read pointer to the location in the file where the next read is to take place.

Because a process is usually either reading from or writing to a file, the current operation location can be kept as pre-process **current-file-position** pointer.

Both the read and write operations use this same pointer, saving space and reducing system complexity.

Repositioning within a file

The directory is searched for the appropriate entry, and the current-file-position pointer is repositioned to a given value. Repositioning within a file need not involve any actual I/O. This file operation is also known as a file seek.

Deleting a file

To delete a file, we search the directory for the named file.

Having found the associated directory entry, we release all file space, so that it can be reused by other files, and erase the directory entry.

Truncating a file

The user may want to erase the contents of a file but keep its attributes.

Rather than forcing the user to delete the file and then recreate it, this function allows all attributes to remain unchanged (except for file length) but lets the file be reset to length zero and its file space released.

Open a file

Before using a file, it must be opened. File attributes and data contents are fetched in main memory for rapid access on later calls.

Memory space in main memory is allocated to store fetched information.

Close a file

When use of file is finished, it should be closed to free up main memory space.

File attributes and data contents are stored back on disk. This may contain modified information if file is updated.

Append

This is a restricted form of write operation. Here, data are only added to the end of a file.

Get Attributes

This operation is used to retrieve file attributes.

Set Attributes

This operation is used to write file attributes. Generally, it is used to change file attributes such as file protection information.

Rename

This operation is used to change the name of an existing file. This is not strictly necessary because a file can be copied to a new file with new name and then old file can be deleted.

File types

- The types of files recognized by the system are **regular**, **directory**, or **special**. However, the operating system uses many variations of these basic types.

Regular files

- Regular files are the most common files and are used to contain data. Regular files are in the form of text files or binary files:

✓ **Text files**

- Text files are regular files that contain information stored in ASCII format text and are readable by the user. You can display and print these files.
- The lines of a text file must not contain NUL characters, and none can exceed bytes in length, including the newline character.

✓ **Binary files**

- Binary files are regular files that contain information readable by the computer.
- Binary files might be executable files that instruct the system to accomplish job. Commands and programs are stored in executable, binary files.
- Special compiling programs translate ASCII text into binary code.

Directory files

- A directory file contains information that the system needs to access all types of files, but directory files do not contain the actual file data.
- As a result, directories occupy less space than a regular file and give the file system structure flexibility and depth.
- Each directory entry represents either a file or a subdirectory.

Special files

- Special files define devices for the system or are temporary files created by processes.
- The basic types of special files are FIFO (first-in, first-out), block, and character
- Windows (and some other systems) use special file extensions to indicate the type of each file**:**

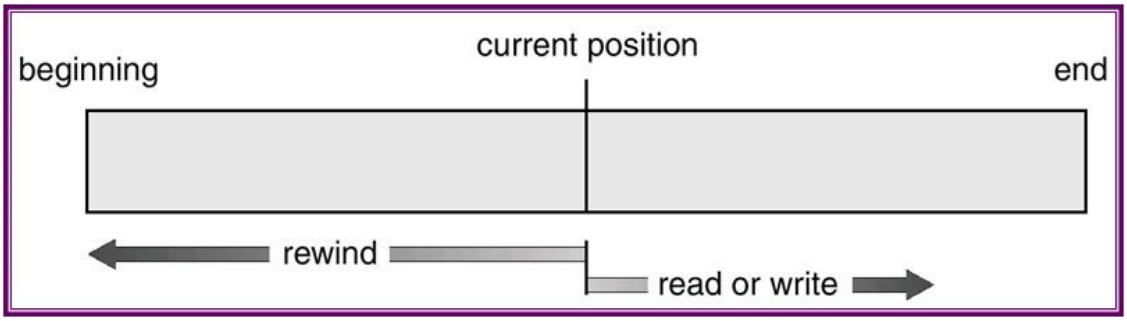
File type	Usual extension	Function
Executable	Exe, com, bin or none	Ready-to- machine-language program
Object	Obj , o	Compiled, machine languagem not linked

Source code	C, cc, java, pas, asm, a	Source code in various languages
Batch	Bat, sh	Commands to the command interpreter
Text	Txt, doc	Textual data, documents
Word processor	Wp, tex, rtf, doc	Various word-processor formats
Library	Lib, a, so, dll	Libraries of routines for programmers
Print or view	Ps, pdf, tar	ASCII or binary file in a format for printing or viewing
Archive	Arc, zip, tar	Related files grouped into one file, sometimes compressed, for archiving or storage
Multimedia	Mpeg, mov, rmm, mp3, avi	Binary file containing audio or A/V information

2.3 File Access Methods

Files store information. When it is used, this information must be accessed and read into computer memory. The information in the file can be accessed in several ways.

Sequential Access



- The simplest access method is sequential access.
- All peripheral devices allow files to be processes sequentially; you start at the beginning of the file and work through each record in turn.
- Sequential files behave as if there were a pointer attached to the file which always indicates the next record to be transferred.
- On devices such as terminals and printers you can only read or write in strict sequential order, but when a file is stored on disc or tape it is possible to use the

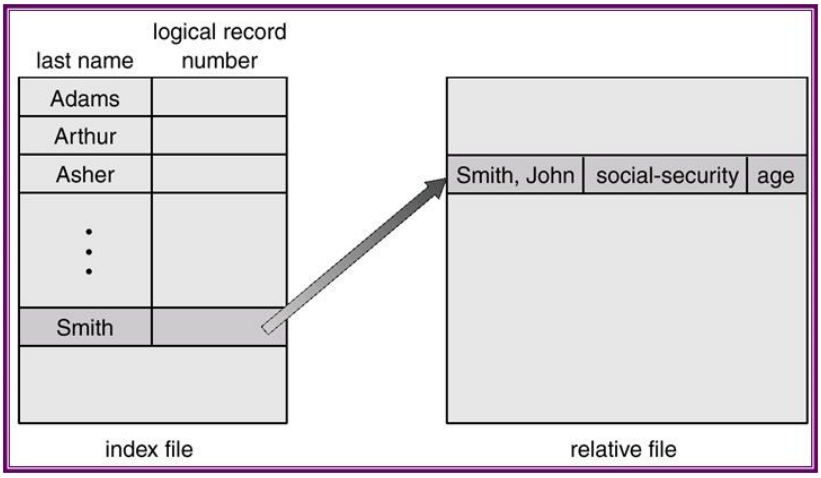
REWIND statement to reset this pointer to the start of the file, allowing it to be read in again or re-written.

Directs Access / Random Access / Relative Access

- Direct-access methods allow records to be read written in any order. Thus, we may read block 14, then read block 53, and then write block 7.
- Most systems only permit this for files stored on random-access devices such as discs; it is sometimes also permitted on tapes.
- All records in a directs-access file must be the same length so that the system can compute the location of a record from its record number.
- The record length has to be chosen when the file is created and (on most systems) is then fixed for the life of the files.
- Direct-access files are of great use for immediate access to large amounts of information. Databases are often of this type.
- For the direct-access method, the file operations must be modified to include the block number as a parameter.
- The block number provided by the user to the OS is normally a relative block number.

  A relative block number is an index relative to the beginning of the file.
  - ✓ Thus, the first relative block of the file is 0, the next is 1, and so on, even though the actual absolute disk address of the block may be 14703 for the first block and 3192 for the second.
- The use of relative block numbers allows the OS to decide where the file should be placed and helps to prevent the user from accessing portion of the file system that may not be part of her file.
- We can easily simulate sequential access on a direct-access file. But, simulating a direct-access file on a sequential-access file is extremely inefficient.

Indexed Access



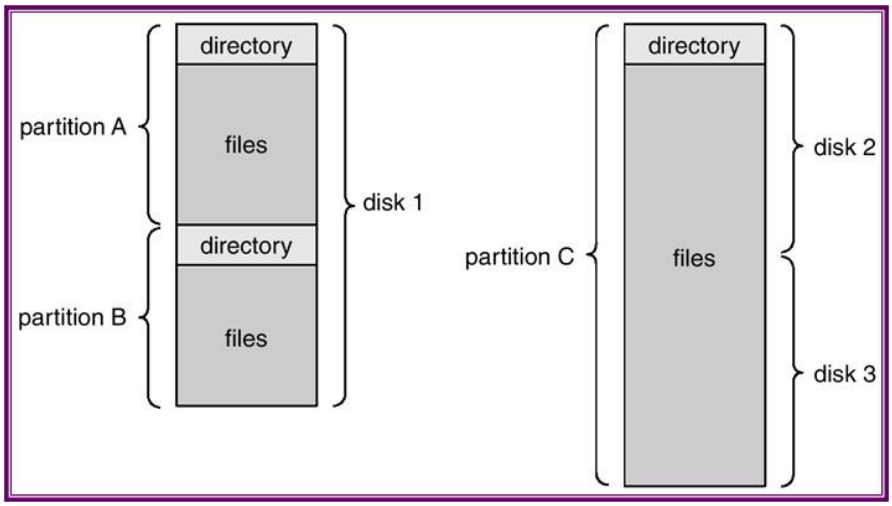
Example of index and relative files.

- These methods generally involve the construction of an index for the file.

  - ✓ To find a record in the file, we first search the index to learn exactly which block contains the desired record.
  - ✓ And then use the pointer to access the file directly and to find the desired record.
- This structure allows us to search a large file doing little I/O.
- With large files, the index file itself may become too large to be kept in memory.
- One solution is to create an index for the index file.
- The primary index file would contain pointers to secondary index files, which would point to the actual data items.
- For example, IBM's indexed sequential-access method (ISAM) uses a small master index that point to disk blocks of a secondary index. The file is kept sorted on a defined key.
- To find a particular item, we first make a binary search of the master index, which provides the block number of the secondary index. The secondary index blocks point to the actual file blocks.
- This block is read in, and again a binary search is used to find the block containing the desired record. Finally, this block is searched sequentially.
- In this way any record can be located from its key by at most to direct-access reads.

2.4 Directory Systems

- Sometimes, it is desirable to place multiple file system on a disk or to use parts of a disk for a file system and other parts for other things, such as swap space or unformatted (raw) disk space.
- These parts are known variously as partitions, slices, or (in the IBM world) minidisks.
- A file system can be created on each of these parts of the disk. We simply refer to a chunk of storage that holds a file system 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 entire in a device directory or volume table of contents.
- The device directory (more commonly known simply as a directory) records information-such as name, location, size, and type-for all files on that volume.

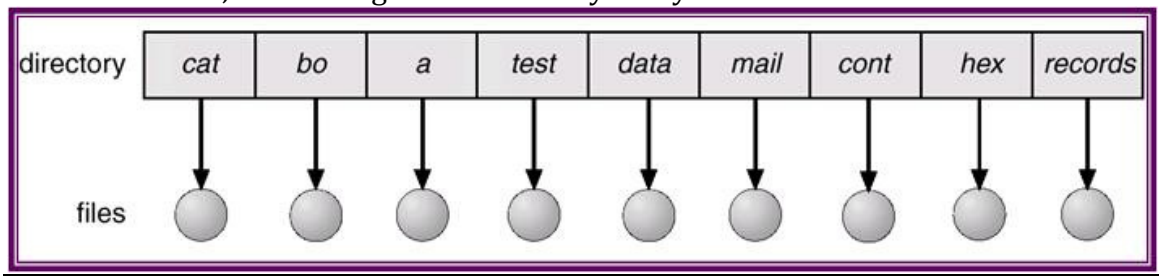


A typical file system organization.

- To keep track of files, file systems normally have directories or folders. Usually, a directory is itself a file.
- The directory can be viewed as a symbol table that translates file names into their directory entries.
- A typical directory entry contains information (attributes, location, and ownership) about a file.
- We went to be able.
  - ✓ To insert entries,
  - ✓ To delete entries,
  - ✓ To search for a named entry,
  - ✓ To list all the entries in the directory.
- When considering a particular directory structure, we need to keep in mind the operations that are to be performed on a directory:
  - ✓ **Search for a file:** we need to be able to search a directory structure to find the entry for a particular file.
  - ✓ **Create a file:** new files need to be created and added to the directory.
  - ✓ **Delete a file:** when a file is no longer needed, we want to be able to remove it from the directory.
  - ✓ **List a directory:** we need to be able to list the files in a directory and the contents of the directory entry for each file in the list.
  - ✓ **Rename a file:** because the name of a file represents its contents to its users, we must be able to change the name when the contents or use of the file changes.
  - ✓ **Traverse the file system:** we may wish to access every directory and every file within a directory structure: for reliability, it is a good idea to save the contents and structure of the entire file system at regular intervals (backup copy).

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.
- On early personal computers, this system was common, in part because there was only one user.
- The world's first supercomputer, the CDC 6600, also had only a single directory for all files, even though it was used by many users at once.



Single – level directories

- A single – level directory has significant limitations, 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. If two users call their data file **test** then the unique – name rule is violated.
- Even a single user on a single – level directory may find it difficult to remember the names of all the files as the number of files increases.

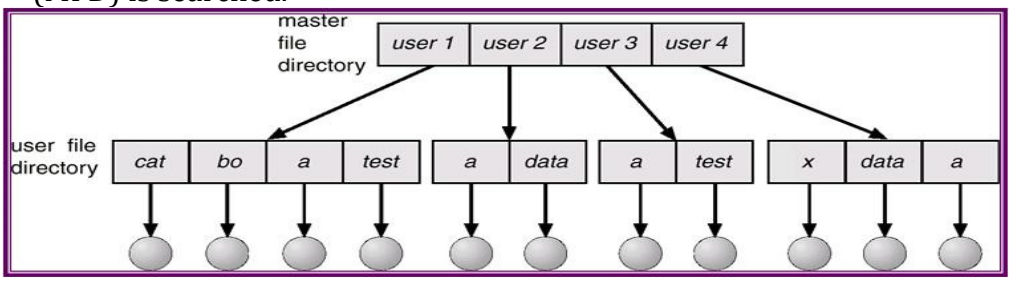
Advantages: This type of structure is very simple. As there is only one directory, searching a file is very quick.

Disadvantage:

- It is not suitable for multi-user systems. Different users may provide same file name for different files, possibly overwriting other's files. This is called a file name collision.
- It is even not suitable for single user system when number of file becomes too large. User can't remember the names of all files.
- Different files can't be grouped.

Two level directory

- The standard solution to limitations of single – level directory is to create a separate directory for each user.
- In the two – level directory structure, each user has his own user file directory (UFD). The UFDs have similar structures, but each lists only the files of a single user.
- When a user job starts or a user logs in, the system's master file directory (MFD) is searched.



Two-level directory Structure

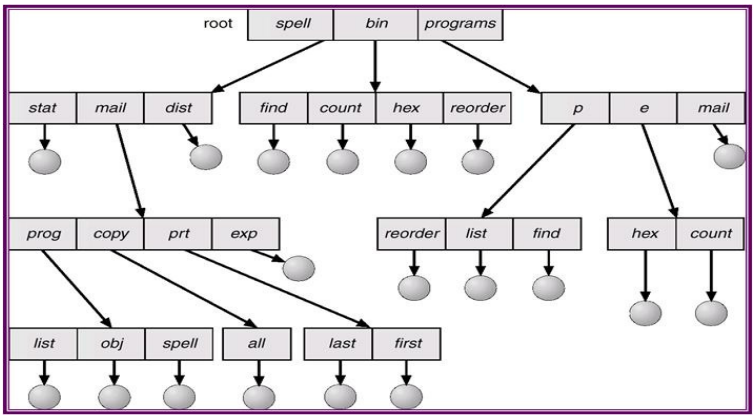
- The MFD is indexed by user name or account number, and each entry points to the UFD for that user (see Fig.10.8).
- When a user refers to a particular file, only his own UFD is searched.
- Although the two-level directory structure solves the name-collision problem, it still has disadvantages.
- This structure effectively isolates one user from another.
- Isolation is an advantage when the users are completely independent but is a disadvantage when the users want to cooperate on some task and to access one another's files.
- A two-level directory can be thought of as a tree, or an inverted 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.
- Specifying a user name and a file name defines a path in the tree from the root (the MFD) to a leaf (the specified file).
- Thus, a user name and a file name define a ***path*** name. To name a file uniquely, a user must know the path name of the file desired.
- Additional syntax is needed to specify the volume of a file. For instance, in MS-DOS a volume is specified by a letter followed by a colon. Thus, a file specification might be c:\userb\test

Advantages: It avoids file name collisions. It can be used in multi-user system. Searching is efficient here; as user has to search in its own single directory.

Disadvantage:

- It is not suitable for users having large number of files.
- Different files cannot be grouped.
- One extra system directory is required to store system files.

Tree Structured Directory



Tree-structure directory structure

- Once we have seen how to view a two-level directory as a two-level tree, the natural generalization is to extend the 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 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),
  - ✓ as a subdirectory (1).
- Path names can be of two types: absolute and relative
  - ✓ 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.
- With a tree-structured directory system, users can be allowed to access, in addition to their files, the files of other users.
  - ✓ For example, user ***B*** can access a file of user ***A*** by specifying its path names.
  - ✓ User ***B*** can specify either an absolute or a relative path name.
  - ✓ Alternatively, user ***B*** can change her current directory to be user ***A***'s directory and access the file by its file names.
  - ✓ Some systems also allow users to define their own search paths.

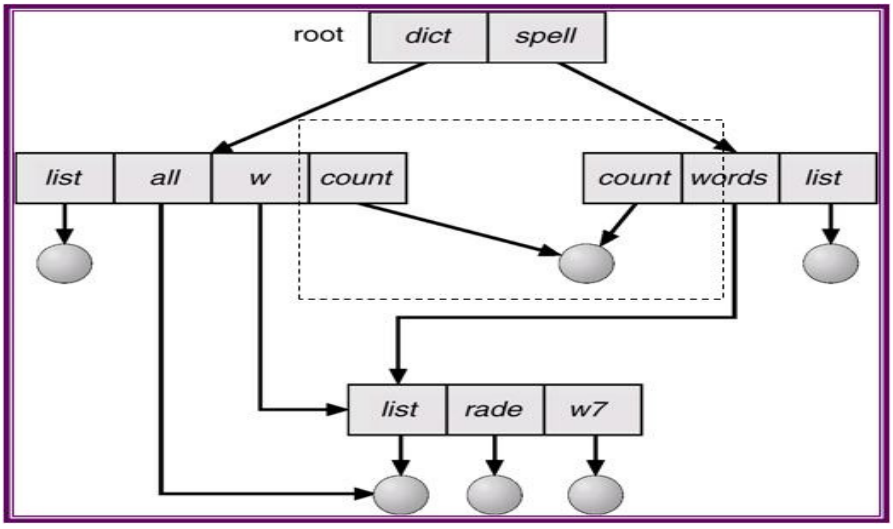
- ✓ In this case, user B could define her search path to be (1) he local directory, (2) the system file directory, and (3) user A's directory, in that order.
- ✓ As long as the name of a file of user A did not conflict with the name of a local file or system file, it could be referred to simply by its name.

Advantages: Different files can be grouped here. It avoids file name collisions. It can be used in multi-user system.

Disadvantage:

- Sharing of files and directories is not possible here.

Acyclic-Graph Directory



Acyclic-graph directory structure.

- It permits users to create shared files and directory.
- Shared file and directories can be created using 'link' operation. This operation allows a file or directory to appear in more than one directory.
- Directories can contain files as well as sub-directories also.
- A file may contain more than one file path.
- Directory structure does not contain cycles.
- In this figure, directories are represented using rectangles, while files are represented using circles. Also figure shows that a file named 'count' is shared from two different locations (directories) 'dict' and 'spell'.

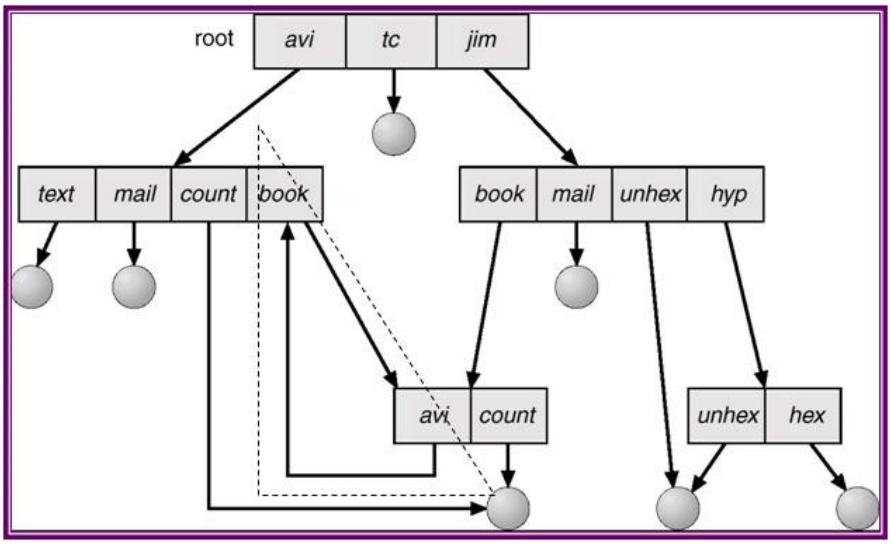
Advantages: It allows sharing of files and directories.

Disadvantage:

- Deletion of shared file or directory is complex. When such file/directory is deleted, directory entry from all related directories should be deleted.

- A file may have multiple file paths. This may create problem in some operations. for example, if user copies al files to backup storage, then such files will be copied multiple times.
- It is difficult to ensure that there are no any cycles in a directory structure.

General-Graph Directory



- It is same as acyclic graph dir, but it also allows cycles in directory structure.
- **Advantages:** It allows sharing of files and directories.
- **Disadvantage:** Searching should be done carefully. Else, it may get into infinite loops.

File Paths

"A file path is a character string divided into separate components by special character such as slash (/).

It is used to uniquely specify a location of a file or directory in a directory structure.

A file path is also called a path name. here, separate components are directories; except that the last component can be a file also.

Different files may contain same file names, but file paths will be unique per files, generally, each file will have single file path. But file can have more than one file path, if it is being shared from two different locations.

Special character which are used to separate different components depends on the operating systems. For example, UNIX uses slash (/) while windows uses backward slash(\) for the same purpose.

Example:

1. c:\progs\cpp\test.cpp --- on windows environment
2. /user/home/test.c ---on UNIX environment

There are two types of defining file paths:

1. **Absolute File-Paths:**
   - It starts with the root directory(or drive name) of a file system.

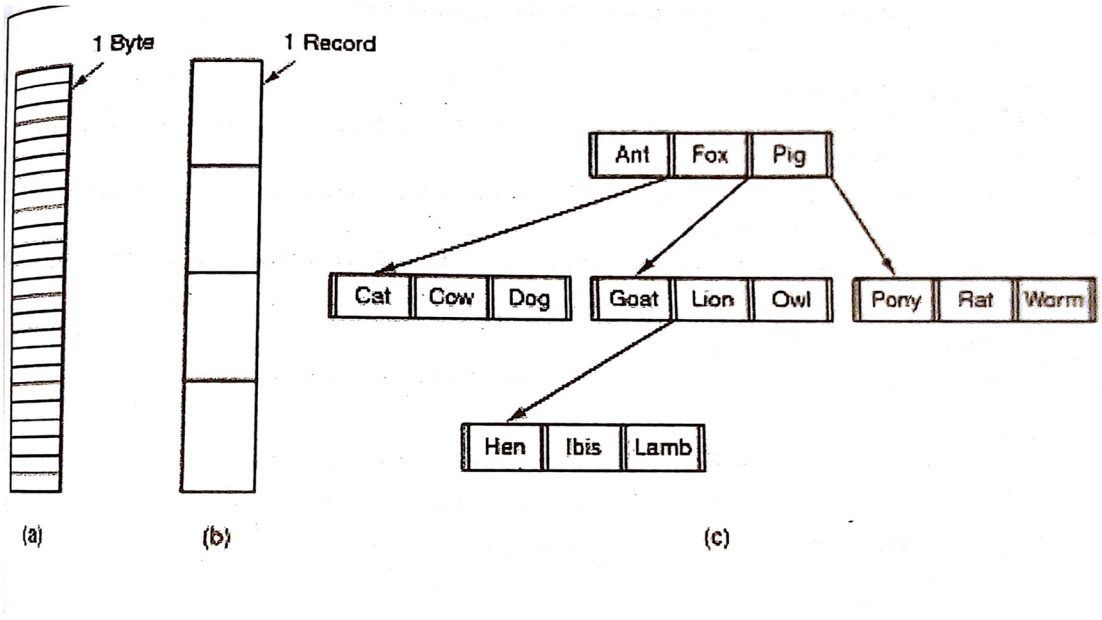
- Here, file paths are given with respect to the root directory.
- C:\progs\cpp\test.cpp **OR** /user/home/test.c

2. Relative File-Paths:

- It starts with the current directory (or current working directory) designated by a user.
- Here file paths are given with respect to such current directories.
- Dot (.) and dotdot (..) can be used as a current directory.
- Example are: consider that two files test.cpp and test.c contains absolute file path as given above.
- If c:\progs is a current directory then path for file test.cpp can be given as cpp\test.cpp
- If /user/home/ is current directory then file test.c can be accessed directly by using its name only test.c or as ./test.c (dot represents current directory).

2.5 File Structures

- "File structure is the way of storing data contents (information) in a file."
- It represents how data are stored in the file.
- There are three common possibilities of storing data in a file. Or, in other words there are three common file structures
  1. Byte Sequence
  2. Record Sequence
  3. Tree
- All these three structures are depicted in following figure



File structure: (a) Byte sequence, (b) Record Sequence, (c) Tree

The description about all these three file structures is as follows:

Byte Sequence

- Data are stored as an unstructured sequence of bytes.
- Operating system does not know or care what is in the file, Entire file is treated as an array of bytes.
- Any meaning of data stored in file can be imposed by user programs,
- Generally, Read operation returns one byte, while write operation overwrite or appends one byte.
- It is a most flexible form of file structures.
- Both UNIX and Windows use this approach.

Record Sequence

- Data are stored as a sequence of fixed length records. Each record will have its own structure.
- Read operation returns one record, while write operation overwrites or appends one record.
- It is suitable for special applications such as database related application.

Tree

- Data are stored as a tree of records. Each record will have its own structure.
- Records may be of varying lengths.
- Each record has key in fixed position in record.
- Records are sorted by key for easy search and retrieval.
- Used with large mainframe systems.

2.6 File Security and Protection Mechanism

- Files are used to store information.
- Depending upon the users and applications, this information can be very important. Consider the importance of various records stored in files in some bank. What will be the consequences if this information gets lost, or damaged or corrupted? So, in short, no one wants to loose information stored in files in any way. Every one wants information stored in files to be safe.
- Here, safety comes in two ways:
    1. Reliability, and
    2. Protection.
- Both of these ways are described below.

Reliability:

- Reliability means safety from physical damage.
- File systems can be damaged by hardware problems, first, dirt, power failures, head crashes and so on. Files may be deleted accidentally (or even intentionally too…!!!).
- Reliability is generally provided by keeping more than one copies of files, means duplicating files.
- Files are duplicated mostly on some different locations, or in different devices. Magnetic tapes are most widely used for backups of large files.
- Operation of duplicating files can be done either automatically using software or manually.
- If original files are damaged (unfortunately), then they can be retrieved back from the backups.

Protection

- Protection means safety from improper access.
- Protection is required where files can be Shared among various users (or even among processes and machines on network).
- Some files need to be shared among all users, while some need to be shared among limited users.
- For example, a user may want to read, write and execute its own Programs. He also wants to share such programs with his partners. But, he doesn't want that other classmates can access these programs.
- The solution to this is to allow limited sharing. This can be done by limiting the type of file access. Here, accessing a file means to perform some operation on that file. Several operations on files can be controlled.
- Some of these operations are:
  - ➔Read. Read a file.
  - ➔Write. Write a file.
  - ➔ Execute. Load and execute a file.
  - ➔ Append: Add information at the end of a file.
  - ➔Delete. Free the space allocated to a file.
- There are two most common Ways to limit the type of file access:
  1. Password
  2. Access Control.
- These two ways are described below:

Password:

- Here, a password is associated with each file. Users can access a file, only if he knows the password
- This scheme looks good, but it suffers from several disadvantages.
- There may be large number f files in a computer system. So user need to remember lots of password.
- To avoid this problem, if single password is used for all files, then once it is discovered all files are accessible.

Access Control:

- Here a list called an access control list (ACL), is associated with each file. This list specifies the user name and type of access allowed for that user. This information is stored for all the possible users of the system.
- This list is generally kept in directory entry along with file name and other information.
- This method also suffers from some problems.
- First problem with this method is: Constructing an ACL is tedious task. Also all users of the system are not known in advance.

- Second problem with this method is: ACL Is kept in directory entry. So, now directory entry should be of varying length instead f fixed length. This results in more complicated space management.
- Solution to this problem is to divide all users in various categories and then to allow access on such categories instead of individual users.
- UNIX uses such type of scheme.
- It divides users in three categories:
    i) Owner,
    ii) Group and
    iii) Others
- Access is given on such category. All the users of that category will be allowed that access.