6... File Management

Chapter Outcomes...

- Explain structure of the given file system with example.
- Describe mechanism of the given file access method.
- Explain procedure to create and access directories and assign the given files access permissions.
- Explain features of the given RAID level structure of hard disk.

Learning Objectives...

- To understand Basic Concepts of File and File System
- To learn various File Access Methods and File Allocation Methods
- To study Directory and its different Structures
- To become familiar with Disk Structure
- To learn Concept of RAID and its Levels

6.0 INTRODUCTION

- File management is one of the important functions of an operating system. It facilitates storage, access and retrieval of both data and programs or all the operating system and of the users.
- File management is the process of manipulating files in a computer system. The operating system defines a logical storage unit known as a file, with all data being stored in the form of files.
- A file is a collection of specific information stored in the memory of the computer system. File management includes the process of creating, modifying and deleting the files.
- File management system is a system software that is concerned with file services such as accessing a file, directory maintenance, access control and others.
- A file management system is that set of system software that provides services to users and applications in the use of files. The system that an operating system or program uses to organize and keep track of files known as file system.
- Some of the **tasks performed by the file management function of operating system** are as follows:
  1. File management helps in creating new files and placing them at a specific location.
  2. It provides a uniform logical view of data to the users rather than physical view, i.e., the internal structure by giving user-friendly interface.
  3. It provides security measures for confidential data, such as electronic funds or criminal records.
  4. File management helps in easily and quickly locating the files in the computer system.
  5. It makes the process of sharing the files among different users easy.
  6. It controls transferring of data blocks between the secondary storage and main memory, as well as between different files.
  7. It helps store the files in separate folders known as directories that ensure better organization of data.
  8. Memory space management is an important function of file management systems and it also allocates and manages space for the files.
  9. File management helps modify the content as well as the name of the file as per the user's requirement.

6.1 FILE (S-16, 18)

- A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks.
- Commonly files represent programs (source and object forms) and data. Data files may be numeric alphabetic, alphanumeric or binary.
- In general a file is a sequence of bits, bytes lines or records whose meaning is defined by the file's creator and user. The information in a file is defined by its creator.
- 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 according to its type. A **text file** is a sequence of characters organized into lines. A **source file** is a sequence of subroutines and functions, 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 code sections that the loader can bring into memory and execute.

6.1.1 Concept (S-16, 18)

- The operating system extracts from the physical properties of its storage devices to define a logical storage unit called as file. Files are mapped by the operating system onto the physical devices.
- The information in a file is defined by its creator. Many different types of information may be stored in a file like 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 according to its type.
- A text file is a sequence of characters organized into lines; a source file is a sequence of subroutines and functions, 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 code sections that the loader can bring into memory and execute. File is defined as "collection of related data. **OR** A file is "collection of data or information".

6.1.2 File Attributes (S-18, 19, W-16, 17)

- A file has certain attributes, which vary from one operating system to another. The Attributes are nothing but the properties of files which store the file related data.
- Attributes are useful to get the information about a particular file or directory. Following is the list of some file attributes.
- File attributes are required by a file system to manage a file. Attributes are useful to get the information about a particular file or directory.
- Following is the list of some common file attributes:
  1. **Name:** The symbolic file name is the only information kept in human readable form.
  2. **Type:** This information is needed for those systems that support different types.
  3. **Location:** This information is a pointer to a device and to the location of the file on that device.
  4. **Size:** The current size of the file (in bytes, words or blocks) and possibly the maximum allowed size are included in this attribute.
  5. **Protection:** Access control information controls that who can do reading, writing, executing and so on.
  6. **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.
  7. **Identifier:** File system gives a unique tag or number that identifies file within file system and which is used to refer files internally.
  8. **Creator or Owner:** A creator is a user or a person who has created that file and the owner is a person who owns that file currently.
- The information about all files is stored in the directory structure, which is present on secondary storage. By using the name of file and its unique identifier we can search the directory for the entry of a file.

6.1.3 File Operations

- 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 operating system provides system calls to create, write, read, reposition, delete and truncate files.
- Files exist to store information and allow it to be retrieved later. Different systems provide different operations to allow storage and retrieval. Common file operations are as follows:

1. **Creating a File:** Two steps are necessary to create a file. First space in the file system must be found for the file. Second an entry for the new file must be made in the directory. The directory entry records the name of the file and the location in the file system.
2. **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. Given the name of the file, the system searches the directory to find the location of file then the write pointer must be updated whenever a write occurs
3. **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. System needs to keep a read pointer to location in the file where the next read is to take place. Once the read has taken place, the read pointer is updated.
4. **Repositioning within a File:** The directory is searched for the appropriate entry, and the current file position is set to a given value. Repositioning within a file does not need to involve any actual I/O. This file operation is also known as a file seeks.
5. **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 and erase the directory entry.
6. **Truncating a File:** Instead of deleting a file and then recreate it, this function allows all attributes to remain unchanged but for the file to be reset to length zero. User wants to erase the contents of the file.

- Other common operations include appending new information to the end of an existing file, and renaming an existing file.

6.1.4 File Types

- To operate on the file in reasonable way an operating system recognizes the type of a file. File naming is given for the easy human interactions or convenience.
- The file name is split into two parts a name of file and extension. For example "student.xls" the file name is student and it is created in MS-Excel application.
- The system uses the extension to indicate the type of the file and the type of operations that can be done on that file. Only a file with a .com, .exe, or .bat extension can be executed, for instance.
- The .com and .exe files are two forms of binary executable files, whereas a .bat file is a containing, in ASCII format, commands to the operating system.
- The following table indicates the common file types.

File Type	Usual Extension	Function
Executable	exe, com, bin or none	Ready to run machine language program.
Object	obj, o	Compiled, machine language, not linked.
Source Code	c, cc, pas, asm, f77	Source code in various languages.
Text	txt, doc	Textual data documents.
Batch	bat, sh	Commands to the command interpreter.
Word Processor	wp, rtf, tex, doc, etc.	Various word processor formats.
Library	lib, a, dll	Libraries of routines for programmers.
Print or View	ps, gif, dvi, pdf	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	Avi, mp3, mov, mkv, mpeg, rm	Binary file containing audio or audio/video information.

6.2 ACCESS METHODS

(S-17, 19, W-18)

- File stores information. When it is used, this information must be accessed and read into computer memory. There are several ways that the information in the file can be accessed.
- An access method describes the manner and mechanisms by which a process accesses the data/information in a file.
- Some system provide only one access method for files while other system, such as those of IBM, support many access method, and choosing the right one for a particular application is a major design problem.

6.2.1 Serial File

(S-17, 19)

- The least complicated form of file organization is the serial file or pile. Data are collected in the order in which they arrive.
- The purpose of this file is simply accumulating the mass of data and save it. Records may have different fields, or similar fields in different orders. Thus, each field should be self-describing including a field name as well as a value.

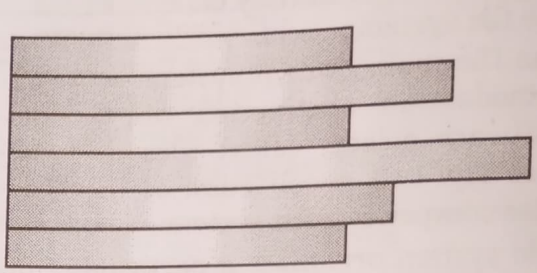


Fig. 6.5: Variable Length Records, Variable Set Of Fields, Chronological Order

Advantages of Serial File:

1. Simple organization.
2. Data usually stored prior to processing.
3. Less complexity and good efficiency for variable sized record.
4. Utilizes space very well for varying data structure.

Disadvantages of Serial File:

1. Because there is no structure to these file, record access is very difficult.
2. Required more searching time.
3. Records are not arrange in proper manner.

6.2.2 Sequential File Access

(S-16, 17, 19, W-16, 18)

- The simplest access method is sequential access. Information in the file is processed in order, one record after the other.
- The bulk of the operations on a file are reads and writes. The read and write operations on the sequential file are done in sequential order.
- A read operation reads the next portion of the file and automatically advances a file pointer, which tracks the I/O location. Similarly a write appends to the end of the file and advances to the end of the newly written material.
- Such a file can be reset to the beginning and on some systems a program may be able to skip forward or backward 'n' records for some integer 'n'.

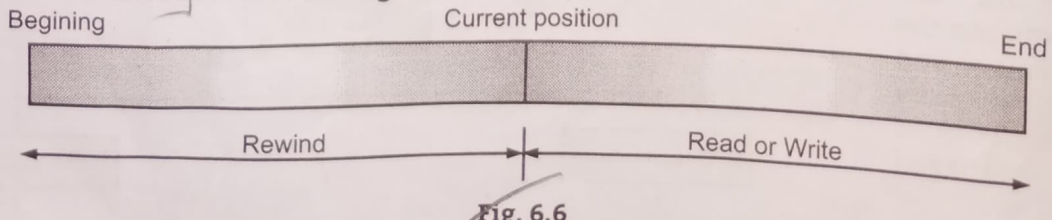


Fig. 6.6

- In this type of file, a fixed format is used for records. All records are of the same length consisting of the same number of fixed length fields in a particular order.
- One particular field usually the first field in each record is referred to as the key field. The key field uniquely identifies the record.

Roll No.	Name	Marks

- Fixed length records, fixed set of fields in fixed order, sequential order based on key field.

Advantages of Sequential File:

1. Easy to access the next record.
2. Data organization is very simple.
3. Absence of data structures.
4. They are easily stored on tapes as well as disks.
5. Automatic backup copy is created.

Disadvantages of Sequential File:

1. Wastage of memory space because of master file and transaction file.
2. For interactive applications that involve queries and/or updates of individual records, the sequential file provides poor performance.
3. It is more time consuming since, reading, writing and searching always start from beginning of file.

- Sequential files are typically used in batch applications where they are involved in the processing of all the records such as payroll, billing etc.

6.2.3 Indexed Sequential File Access

(S-17, 19)

- An indexed sequential file is a sequential file in which the records are indexed. An indexed sequential file is an improvement over a sequential file.
- Two features are added in this file namely, an index to the file to support random access, and an overflow file.
- The index provides a lookup capability to reach quickly the vicinity of a desired record while the overflow file is similar to the log file used with a sequential file but is integrated so that a record in the overflow file is located by following a pointer from its predecessor record.
- Indexing of records provides the facility of searching the records randomly. An indexed file is a simple sequential file that contains an index as its records.
- Entries in indexed files are made up of two fields, the key field, which is the same as the key field in the main file and a pointer pointing to some record in the main file.
- To find a specific field in the main file, the index is searched for the highest key value, which is equivalent to the desired value.
- The pointer related to key field start searching the record at location it indicates.
- The search continues in the main file at the location indicated by the pointer.

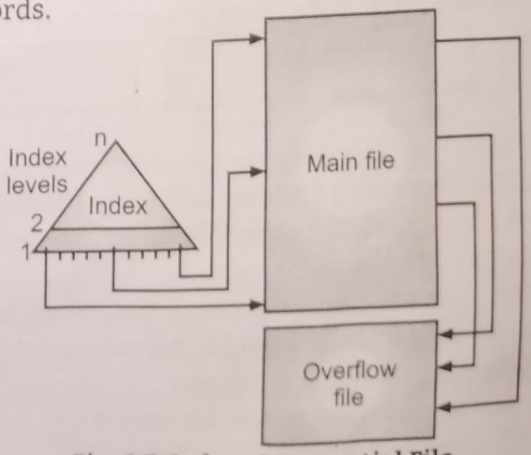


Fig. 6.7: Indexed Sequential File

Advantages of Index Sequential File:

1. Variable length records are allowed.
2. Indexed sequential file may be updated in sequential or random mode.
3. Very fast operation.

Disadvantages of Index Sequential File:

1. The major disadvantage of the indexed sequential file is that, as the file grows, a performance deteriorates rapidly because of overflows and consequently there arises the need for periodic reorganization. Reorganization is an expensive process and the file becomes unavailable during reorganization.
2. When a new record is added to the main file, all of the index files must be updated.
3. Consumes large memory space for maintaining index files.

Indexed File Access: (S-17)

- Both the sequential and indexed sequential file searches the records on the basis of key fields. If searching is required on the basis of other attributes, i.e., based on some criteria, such as the value in any field, then both these files become inadequate. The third form of file organization scheme that supports this feature is known as indexed file.
- In indexed file access method, multiple indexes are maintained, one for each field of a record. Records are searched only through these indexes. Whatever may be the field, its pointer should exist in the related index for searching.
- Indexed files are used mostly in applications where timeliness of information is critical and where data are rarely processed exhaustively. Examples are airline reservation systems and inventory control systems.

6.2.4 Direct File Access (S-17, 19, W-18)

- Another method is direct access or relative access. A file is made up of fixed length logical records that allow programs to read and write records rapidly in no particular order.
- The direct file or hashed file exploits the capability found on disks to access directly any block of a known address.
- Direct files are often used where very rapid access is required, where fixed-length records are used, and where records are always accessed one at a time. Examples are directories, pricing tables, schedules, and name lists.
- In direct file access, records are accessed directly by the physical addresses of the location, at which the records are stored in the disk.
- Key field for searching is required here as well; similar to the sequential file, but the concept of ordering records sequentially is not needed here. Records in direct files are of fixed length and are accessed one at a time.
- The direct-access method is based on a disk model of a file, since disks allow random access to any file block. For direct access, the file is viewed as a number of sequential blocks of records.
- A block is generally a fixed length quantity, defined by an operating system. A block may be a 512 bytes long, 1024 bytes long or some other length, depending upon the system.

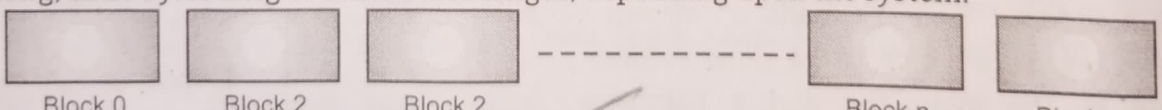


Fig. 6.8

- A direct access file allows arbitrary blocks to be read or written. Thus we may read block 14, then read block 50 and then write block 7. There are no restrictions on the order of reading or writing for a direct access file.
- Direct files are used in those applications, where speed is the main constraint for accessing. Hashing technique is usually used for accessing the records in direct access files.

Advantages of Direct File Access:

1. Using this method we can access any records randomly.
2. It gives fastest retrieval of records.

Disadvantages of Direct File Access:

1. Wastage of storage space, if hashing algorithm is not chosen properly.
2. This method is complex and expensive.

tape - is a device that stores computer data on a magnetic tape

- The three major activities of an operating system in regard to secondary storage management are:
  1. Managing the free space available on the secondary-storage device.
  2. Allocation of storage space when new files have to be written.
  3. Scheduling the requests for memory access.

6.2.6 File Allocation Methods

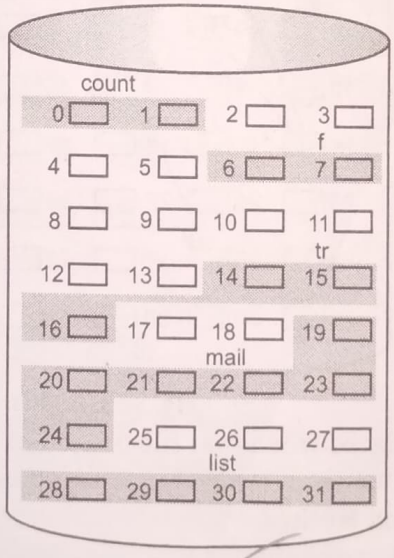
(S-17, 19, W-18)

- From the user's point of view, a file is an abstract data type. It can be created, opened, written, read, closed and deleted without any real concern for its implementation.
- The implementation of a file is a problem for the operating system. On a tape based system we can map each file to a separate tape or several files on to the same tape.
- The main problem is how to allocate space to these files so that disk space is effectively utilized and files can be quickly accessed.
- Files are allocated disk spaces by operating system. The file allocation methods refer to keep track of which file has been allocated to which disk blocks.
- Three major methods of allocating disk space are Contiguous File Allocation, Linked File Allocation and Indexed File Allocation. Each method has its advantages and disadvantages.
- The allocation methods define how the files are stored in the disk blocks. The main idea behind allocation is effective utilization of file space and fast access of the files.

6.2.6.1 Contiguous File Allocation

(W-18, S-19)

- The contiguous file allocation method requires each file to occupy a set of contiguous addresses on the disk. Disk addresses define a linear ordering on the disk.
- Contiguous file allocation of a file is defined by the disk address of the first block and its length. If the file is 'n' blocks long and starts at location 'b', then it occupies blocks b, b+1, b+2, - - - - - b+n-1.
- The directory entry for each file indicates the address of the starting block and the length of the area allocated for this file.



Directory

File	Start	Length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Fig. 6.12: Contiguous Allocation

- Contiguous allocation supports both sequential and direct access. Accessing a file that has been allocated contiguously is easy. For sequential access, the file system remembers the disk address of the last block referenced and, when necessary, reads the next block.
- For direct access to block 'i' of a file, which starts at block 'b', we can immediately access block b+i.
- The difficulty with contiguous allocation is finding space for a new file.
- Once the free space is defined, we can decide how to find space for contiguously allocated file. If file to be created are 'n' blocks long, we must search free space list for 'n' free contiguous blocks.

Advantages of Contiguous File Allocation Method:

1. Supports both sequential and direct access methods.
2. Contiguous allocation is the best form of allocation for sequential files. Multiple blocks can be brought in at a time to improve I/O performance for sequential processing.
3. It is also easy to retrieve a single block from a file. For example, if a file starts at block 'n' and the i^{th} block of the file is wanted, its location on secondary storage is simply $n + i$.
4. Reading all blocks belonging to each file is very fast.
5. Provides good performance.

Disadvantages of Contiguous File Allocation Method:

1. Suffers from external fragmentation.
2. Very difficult to find contiguous blocks of space for new files.
3. Also with pre-allocation, it is necessary to declare the size of the file at the time of creation which many a times is difficult to estimate.
4. Compaction may be required and it can be very expensive.

6.2.6.2 Linked File Allocation (Chained File Allocation) (W-18, S-19)

- Linked allocation solves all problems of contiguous allocation. With linked allocation, each file is a linked list of disk blocks; the disk blocks may be scattered anywhere on the disk.
- The directory contains a pointer to the first (and last) blocks of the file. For example a file of 5 blocks, which starts at block 9 might continue at block 16, then block 1, block 10, and finally block 25. Each block contains a pointer to the next block.
- These pointers are not made available to the user, thus if sector is 512 words and a disk address (the pointer) requires two words, then the user sees blocks of 510 words.
- Creating a file is easy. We simply create a new entry in the device directory. A write to a file removes first free block from free space list and write to it. This new block is then linked to the end of the file.
- To read a file, we simply read block by following the pointers from block to block. There is no external fragmentation with linked allocation, and any free block on the free-space list can be used to satisfy a request.

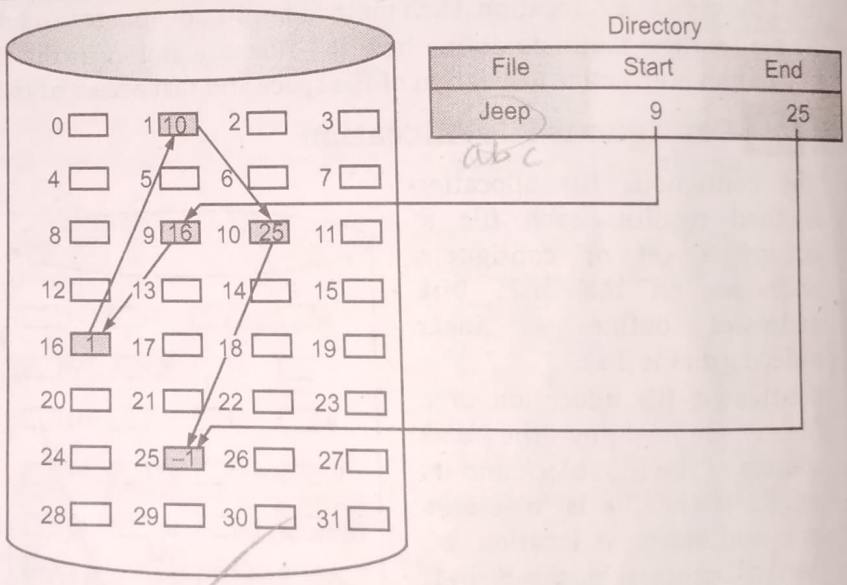


Fig. 6.13: Linked Allocation of Disk Space

- The size of a file need not be declared when that file is created. A file can continue to grow as long as free blocks are available. Consequently, it is never necessary to compact disk space.

Advantages of Linked File Allocation Method:

1. Any free blocks can be added to a chain.
2. There is no external fragmentation.
3. Best suited for sequential files that are to be processed sequentially.
4. No need to know the size of the file in advance.
5. The disk address of first block can be used to locate the rest of the blocks.
6. Never necessary to defragment disk. Blocks are completely utilized here. So no disk fragmentation.
7. No need to compact or relock files.

Disadvantages of Linked File Allocation Method:

1. There is no accommodation of the principle of locality that is series of accesses to different parts of the disk are required.
2. Space is required for the pointers, 1.5% of disk is used for the pointers and not for information. If a pointer is lost or damaged or bug occurs in operating system or disk hardware failure occur, it may result in picking up the wrong pointer.
3. This method cannot support direct access.
4. It is not an efficient scheme because the list traversal needs to read each block which is quite time consuming.

6.2.6.3 Indexed File Allocation (W-18, S-19)

- Linked allocation solves the external fragmentation and size declaration problems of contiguous allocation. However linked allocation cannot support direct access, since the blocks are scattered all over the disk.
- Mostly pointers to blocks are scattered all over the disk. Indexed allocation solves this problem by bringing all of the pointers together into one location the Index Block.
- Each file has its own index block, which is an array of disk block addresses. The ith entry in the index block points to the ith block of the file. The directory contains the address of the index block.
- To read the ith block we use pointer in ith index block entry to find and read the desired block. When the file is created, all pointers in the index block are set to nil. When the ith block is first written a block is removed from the free space list and its address is put in the ith index block entry.
- Indexed allocation supports direct access, without suffering from external fragmentation. Indexed allocation does suffer from wasted space. The pointer overhead of index block is worse than pointer over head of linked allocation.
- Assume we have a file of only one or two blocks with linked allocation we only lose the space of one pointer per block. With indexed allocation an index block must be allocated even if only one or two pointers will be non-nil.

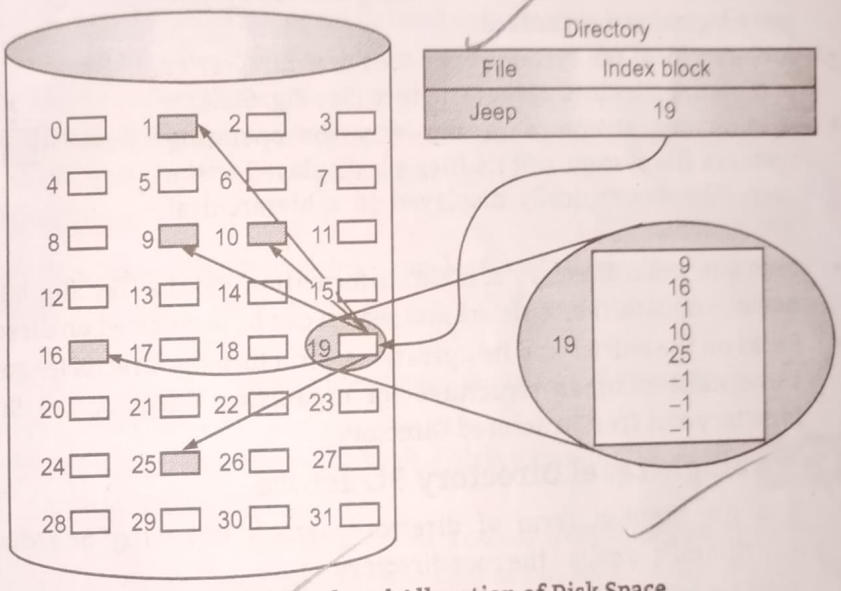


Fig. 6.14: Indexed Allocation of Disk Space

Advantages of Indexed File Allocation Method:

1. Does not suffer from external fragmentation.
2. Support both sequential and direct access to the file.
3. No need for user to know size of the file in advance.
4. Indexing of free space can be done by mean of the bit map.
5. Entire block is available for data as no space is occupied by pointers.

Disadvantages of Indexed File Allocation Method:

1. It required lot of space for keeping pointers so wasted space of memory.
2. Indexed allocation is more complex and time consuming.
3. Overhead of index blocks is not feasible for very small file.
4. Overhead of index blocks is not feasible for very big file also, because it is difficult to manage levels of indices.
5. Keeping index in memory requires space.

6.3 DIRECTORY STRUCTURE (S-19)

- Numbers of files are stored on the disk. To keep track of files, file systems normally have directories or folders, which in many systems are themselves files. To manage all these files, we organized them in to structure called directory structure.
- Directories are basically symbol tables of files. A single flat directory can contain a list of all files in a system. The directory structure is the organization of files into a hierarchy of folders.

6.3.1 Concept of Directory

(S-19)

- To organize files in the computer system, in a systematic manner, the operating system provides the concept of directories. A directory can be defined as a way of grouping files together.
- Millions of files, present in the system, need to be managed. Directories provide the means to organize files in structure. Directory is itself a file that is owned by the operating system.
- The directories are organized in a hierarchical manner, allowing users to create subdirectories under their directories, thus making the file system more logical and organized.
- A hierarchical file system is one that uses directories to organize files into a tree structure (See Fig. 6.15).
- A directory structure is the way an operating system's file system and its files are displayed to the user. Files are typically displayed in a hierarchical tree structure.

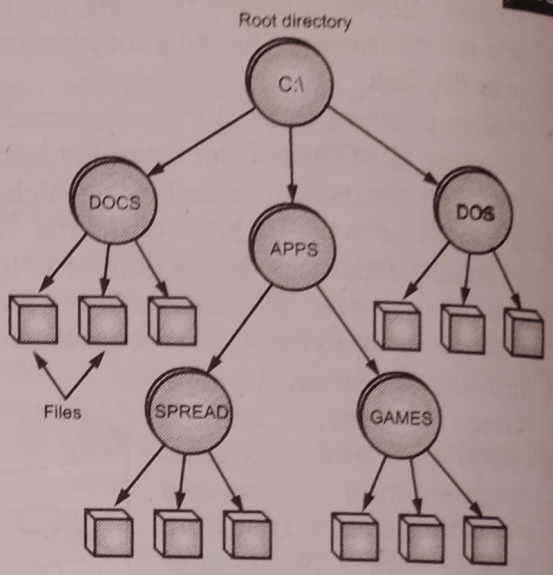


Fig. 6.15

- Each entry in a directory contains information about a file. Similar to file operations such as create, rename, update, insert, delete and search can be performed on directories.
- Based on the entries and its operations, structure for directories can be organized in different ways. Three most common structures for organizing directory are Single Level Directory, Two Level Directory and Tree Structured Directory.

6.3.2 Single Level Directory Structure

(W-16, 18, S-19)

- It is the simplest form of directory system is having one directory containing all the files. Sometimes, it is called the root directory.
- In single level directory structure, the entire files are contained in the same directory. So unique name must be assigned to each file of the directory.
- The single level directory structure appears as the list of files or a sequential file having the file names serving as the key. Single level directory structure was implemented in the older versions of single user systems.
- The world's first supercomputer, the CDC 6600, had only a single directory for all files, even though it was used by many users at once.
- Fig. 6.16 shows the structure of single level directory structure.

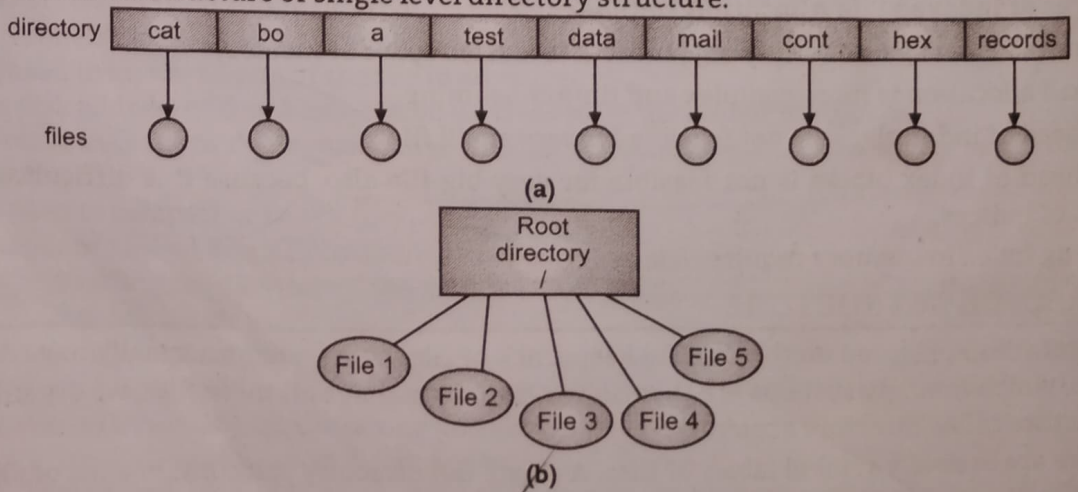


Fig. 6.16: Single Level Directory Structure

Advantages of Single Level Directory Structure:

1. Single level directory structure is easy to implement and maintain.
2. It is simple directory structure.
3. Single level directory structure, the operations like creation, searching, deletion, updating are very easy and faster.

Disadvantages of Single Level Directory Structure:

1. It having only one directory in a system so there may chance of name collision because two files cannot have the same name.
2. In single level directory structure, difficult to keep track of the files, if the number of files increases.
3. This directory is not used on multi-user systems but could be used on a small embedded system.
4. The files such as graphics, text etc. are inconvenient for this data structure.
5. File names are generally selected to reflect the content of the file; they are often limited in length, complicating the task of making file names unique. The MS-DOS operating system allows only 11-character file names; UNIX, in contrast, allows 255 characters.

6.3.3 Two Level Directory Structure (S-17, 18, 19, W-16)

- The structure of two level directory structure is divided into two levels of directories namely, a master directory and user directories. The user directories are the sub-directories of the master directory.
- In two level directory structure, a separate directory is provided to each user and all these directories are contained and indexed in the master directory.
- The user directory represents a list of files of a specific user. In this directory structure, each users has its private directory known as User File Directory (UFD).
- When a user refers to a particular file, only his own UFD is searched. Thus, different users may have files with the same name, as long as all the file names within each UFD are unique. To delete a file, the operating system searches the local UFD of that user.
- The user directories themselves must be created and deleted as necessary. A special system program is run with the appropriate user name and account information. The program creates a new UFD and adds an entry for it to the Master File Directory (MFD). Thus the two level directories solve the name collision problem.
- Fig. 6.17 shows structure of two level directory structure. It looks like an inverted tree. The root is the master directory which having user directories as its branches and files as the leaves of these branches.
- Two level directory structure used on a multi-user computer or on a simple network of personal computers that shared a common file server over a local area network.
- A two-level directory as a tree of height 2:
  - The root of the tree is the MFD.
  - Its direct descendants are the UFDs.
- The two level directory structure is the one most used in UNIX and MS-DOS.

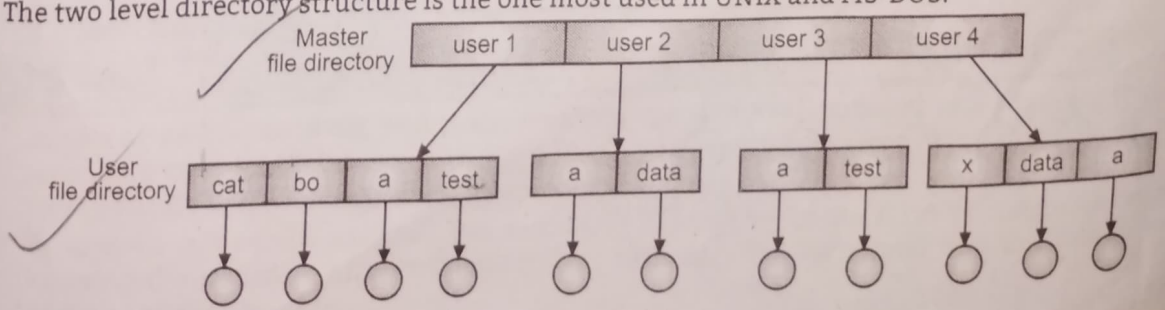
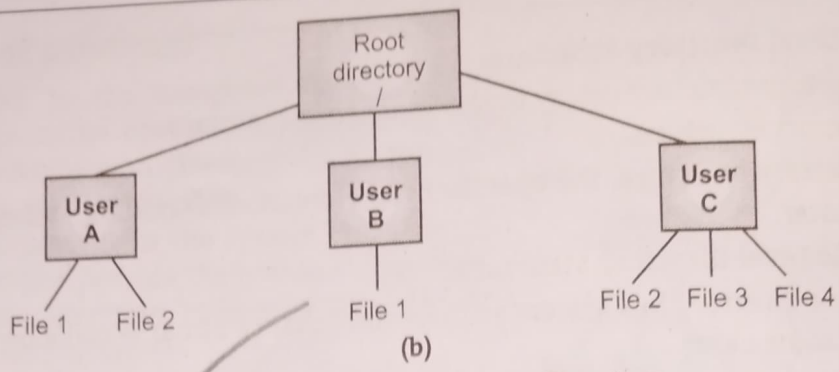


Fig. 6.17 (a)



(b)

Fig. 6.17: Two Level Directory Structure

Advantages of Two Level Directory Structure:

1. It solves the file name collision problem by creating own user directory.
2. This method isolates one user from another and protects user's files.
3. Different users may have files with same name.

Disadvantages of Two level directory

1. Still it not very scalable, two files of the same type cannot be grouped together in the same user.
2. Sharing of files by different users is difficult.

6.3.4 Tree Structured Directory Structure (S-19)

- The two level hierarchies eliminate name conflicts among users but are not satisfactory for users with a large number of files. We needed general hierarchy i.e., a tree of directories.
- The tree structured directory structure, allows users to create their own subdirectory and to organize their files accordingly. A subdirectory contains a set of files or subdirectories. A directory is simply another file, but it is treated in a special way.
- Fig. 6.18 (a) shows tree structured directories. MS-DOS system is a tree structure directory.
- With this approach, each user can have as many directories as are needed so that files can be grouped together in natural ways. Fig. 6.18 (b), shows users A, B, C directories contained in the root directory each belong to a different user.
- The ability for users to create an arbitrary number of subdirectories provides a powerful structuring tool for users to organize their work. Users can access the files of other users.

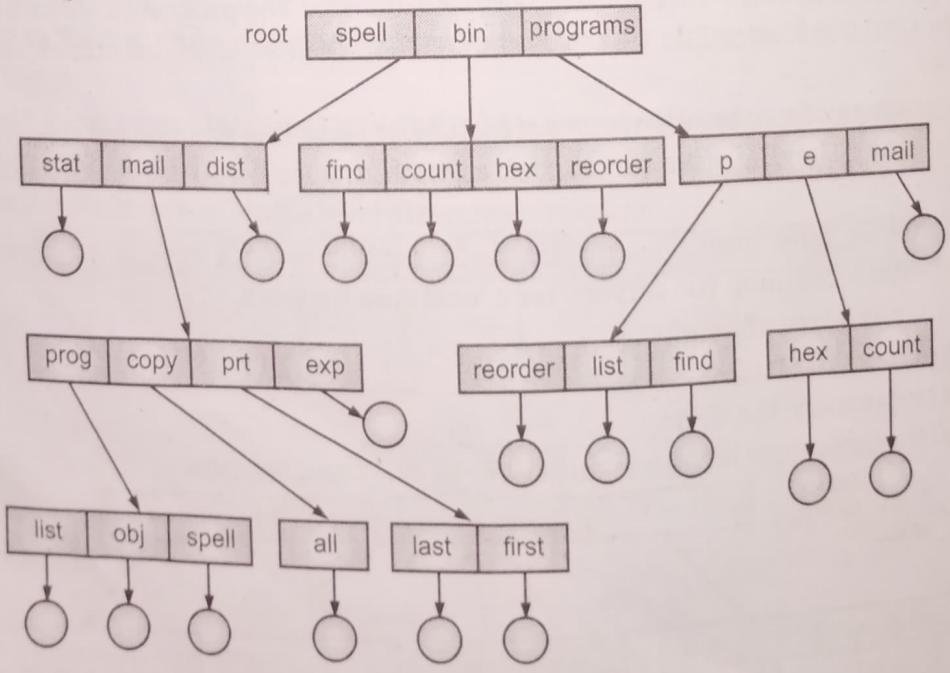
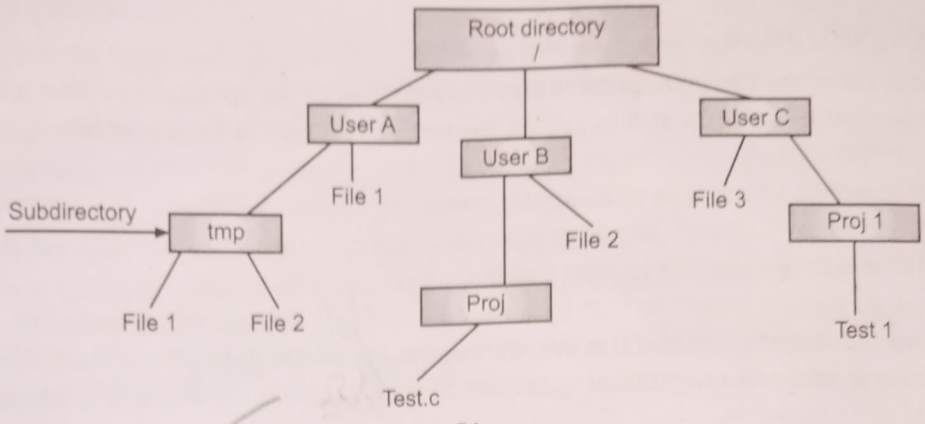


Fig. 6.18 (a)



(b)

Fig. 6.18: Example of Tree Structured Directory Structure

- In this type of a directory structure, each file is identified by a path name. The path name in tree structured directory structure, starts from the root directory. For example, the file Test.c will have a path name /user B/proj/Test.c. Every file has a unique path name.
- A path name is used to change the current directory to the required file or directory. There are two types of path names:
  1. **Absolute Path:** An absolute path begins at the root and follows a path down to the specified file, giving the directory names on the path.
  2. **Relative Path:** A Relative path defines a path from the current directory. If the current directory is root/spell/mail, then the relative path name prt/first refers to the same file as does the absolute path name root/spell/mail/prt/first.
- For example, in Fig. (b), /UserB/proj/Test.c is an absolute file name, whereas proj/Test.c refers to the same file if the current directory is /UserB.
- If a directory is empty, its entry in the directory that contains it can simply be deleted. However, suppose the directory to be deleted is not empty but contains several files or subdirectories.
- One of two approaches can be taken. Some systems, such as MS-DOS, will not delete a directory unless it is empty. Thus, to delete a directory, the user must first delete all the files in that directory.
- If any subdirectories exist, this procedure must be applied recursively to them, so that they can be deleted also. This approach can result in a substantial amount of work.
- An alternative approach, such as that taken by the UNIX rm command, is to provide an option: when a request is made to delete a directory, all that directory's files and subdirectories are also to be deleted.

Advantages of Tree Structured Directory:

1. User can create directory as well as subdirectory.
2. Users can be provided access to a sub directory rather than the entire directory.
3. It provides a better structure to file system.
4. Managing millions of files is easy with tree structured directory.

Disadvantages of Tree Structured Directory:

1. The tree structure can create duplicate copies of the files.
2. The users could not share files or directories.
3. It is inefficient, because accessing a file may go under multiple directories.
4. Search time may become unnecessarily long.

6.4 DISK ORGANIZATION AND DISK STRUCTURE

- In recent years, processor and main memory speeds have increased more rapidly than those of secondary storage devices, such as hard disks. As a result, processes requesting data on secondary storage tend to experience relatively long service delays.
- In modern computers, most of the secondary storage is in the form of magnetic disks. Hence, knowing the structure of a magnetic disk is necessary to understand how the data in the disk is accessed by the computer.