8...

File System

Objectives...

- To introduce concept of file system.
- To know about various file access and allocation methods.
- To get knowledge of file structure.
- To learn free space management techniques.

8.1 INTRODUCTION AND FILE CONCEPTS

- A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks.
- 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.

8.1.1 File Attributes [S-19, 23; W-22]

- A file has certain attributes, which vary from one operating system to another, but typically consist of these:

1. **Name**: Every file carries a name by which the file is recognized in the file system. One directory cannot have two files with the same name.
2. **Identifier**: Along with the name, Each File has its own extension which identifies the type of the file. For example text files have an extension .txt or .doc, spreadsheets have extension .xlsx and audio/video files have an extension .mp3 and .mp4 etc.
3. **Type**: In a File System, the Files are classified in different types such as video files, audio files, text files, executable files, etc.
4. **Location**: In the File System, there are several locations on which, the files can be stored. Each file carries its location as its attribute.

5. **Size**: The Size of the File is one of its most important attribute. By size of the file, we mean the number of bytes acquired by the file in the memory.
6. **Protection**: The Admin of the computer may want the different protections for the different files. Therefore each file carries its own set of permissions to the different group of Users.
7. **Time and Date**: Every file carries a timestamp which contains the time and date on which the file is last modified.

8.1.2 Operations on Files [W-18]

- There are various operations which can be implemented on a file. We will see all of them in detail:

1. **Create**: Creation of the file is the most important operation on the file. Different types of files are created by different methods. For example, text editors are used to create a text file, word processors are used to create a word file and Image editors are used to create the image files.
2. **Write**: Writing the file is different from creating the file. The OS maintains a write pointer for every file which points to the position in the file from which, the data needs to be written.
3. **Read**: Every file is opened in three different modes: Read, Write and Append. A Read pointer is maintained by the OS, pointing to the position up to which, the data has been read.
4. **Re-position**: Re-positioning is simply moving the file pointers forward or backward depending upon the user's requirement. It is also called as **Seeking**.
5. **Delete**: Deleting the file will not only delete all the data stored inside the file. It also deletes all the attributes of the file. The space which is allocated to the file will now become available and can be allocated to the other files.
6. **Truncate**: Truncating is simply deleting the file except deleting attributes. The file is not completely deleted although the information stored inside the file gets replaced.

8.1.3 Types of Files

- It refers to the ability of the operating system to differentiate various types of files like text files, binary, and source files. However, Operating systems like MS-DOS and UNIX has the following type of files:

(i) Character Special File:

- It is a hardware file that reads or writes data character by character, like mouse, printer, and more.

(ii) Ordinary Files:

- These types of files stores user information.
- It may be text, executable programs, and databases.
- It allows the user to perform operations like add, delete, and modify.

(iii) Directory Files:

- Directory contains files and other related information about those files. It is basically a folder to hold and organize multiple files.

(iv) Special Files:

- These files are also called device files. They represent physical devices like printers, disks, networks, flash drive, etc.

Table 8.1: File Types and Functions

File Type	Usual Extensions	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, java, pas, asm, a	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	Various word-processor formats.
Library	lib, a,dll	Libraries of routines for programmers.
Print or View	ps, gif, dvi	ASCII or binary file in a format for printing or viewing.
Multimedia	Mov,mpeg,rm,mp3,avi	Binary file containing audio or A/V information.
Archive	arc, zip, tar	Related files grouped into one file, sometimes compressed, for archiving or storage.

8.1.4 Functions of File

- Create file, find space on disk, and make an entry in the directory.
- Write to file, requires positioning within the file.
- Read from file involves positioning within the file.
- Delete directory entry, regain disk space.
- Reposition: move read/write position.

8.1.5 Commonly used terms in File Systems

- **Field:** This element stores a single value, which can be static or variable length.
- **Database:** Collection of related data is called a database. Relationships among elements of data are explicit.
- **Files:** Files is the collection of similar record which is treated as a single entity.

- **Record:** A Record type is a complex data type that allows the programmer to create a new data type with the desired column structure. It groups one or more columns to form a new data type. These columns will have their own names and data type.

8.2 ACCESS METHODS [S-23]

- There are various types of file access methods in the operating system:
  - Sequential Access Method
  - Direct Access Method
  - Index Sequential Access Method

8.2.1 Sequential Access

- Sequential Access method is one of the simplest file access methods. Most of the OS uses a sequential access method to access the file.
- In this method, the OS read the file word by word and we have a pointer that points the file's base address.
- If we need to read file's first word, then there is a pointer that offers the word in which we want to read and increment value of the word by 1. This process continues till the end of the file.
- The sequential access method is one of the mostly used methods because more files like text files, audio files, and video files require to be sequentially accessed.

Key Points:

- In the sequential access method, if we use read command, then the pointer is moved ahead by 1.
- If the write command is used, then the memory will be allocated, and at the end of the file, a pointer will be moved.

Advantages:

1. Easy to access the next record.
2. Data organization is very simple.
3. Absence of auxiliary data structures.
4. Sequential files are typically used in batch applications where they are involved in the processing of all the records such as payroll, billing etc.
5. Sequential Access Method is reasonable for tape.
6. Automatic backup copy is created.

Disadvantages:

1. Record deletion creates a waste space. It requires some type of record reorganization.
2. The sequential file provides poor performance for interactive applications that involve queries and/or updates of individual records.

8.2.2 Direct Access [S-18]

- 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 512 bytes long, 1024 bytes long or some other length, depending upon the system.
- 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 access files are of great use for immediate access to large amounts of information.
- When a query concerning a particular subject arrives, we compute which block contains the answer and then read the block directly to provide the desired information.
- For example, in an Airline Reservation System, we store all the information about a particular flight in the block identified by the flight by number (flight 710). Thus, the number of available seats for flight 710 is stored in block 710 of the reservation file.
- The block number provided by the user to the operating system is normally 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.
- The use of relative block numbers allows the operating system to decide where the file should be placed.
- Not all the operating system support both sequential and direct access of files. Some systems allow only sequential file access, others allow only direct access.

Advantages of a direct access file:

1. Data access is direct and fast.
2. Centrally maintained data can be kept up-to-date.

Disadvantages of a direct access file:

1. As the entire data is to be stored on disk but disk (hardware) is expensive.
2. There is no backup if a file gets destroyed. This is so because files are updated directly and no transaction files are maintained.

8.2.2.1 Hashing

- The basic idea of Hash addressing is that each record is placed in the database at a location whose Stored Record Address(SRA) may be computed as some function (Called as Hash function) of a value usually the primary key value.
- Thus, to store the record initially, the DBMS computes SRA and instructs the access method to place the occurrence at that position. To retrieve the occurrence, the DBMS performs the same computation as before and then requests the access method to fetch the occurrence at the computed position.
- What address is generated by the Hashing function? There are a number of ways of converting a key to a numeric value. Most of the keys are numeric, but if the keys are alphabetic or alpha numeric we can use the bit representation of the alphabet to generate the numeric equivalent key.

- A number of simple hashing methods are given below:
  1. Use the **low order part of the key**.
  2. For long keys, we identify start, middle and end regions, such that the sum of the lengths of the start and end regions equals the length of the middle region. The start and end digits are combined and the combined string of digits is added to the middle region digits. The (new number) mod (the upper limit of the hash function) gives the **bucket address**.
  3. **Square** all or part of the key and take a part from the result. The whole or some defined part of the key is squared and a number of digits are selected from the square as being part of the hash result.
  4. **Division** can be used to form the address. The key can be divided by a number (consider it as a prime number) and the remainder is taken as the bucket address usually a prime number is used for division because if keys are in some multiples, this would produce a poor result.

Hash Function:

- There are a number of possible methods for generating a hash function but it has been found that hash functions using division or multiplication perform quite well under most conditions.
- We will take a simple hashing function:

$$H(k) = k \bmod s$$

where, H(k) produces an address and H is a hash function that maps the key value k to the value H(k) and k is the numeric representation of the key.

Advantages:

1. This method provides very fast direct access on the basis of values of the hashed field.

Disadvantages:

1. The sequence of stored record occurrence will not be a primary key sequence i.e. it has no particular sequence.
2. The possibility of collision i.e. two distinct stored record occurrences whose keys may hash to the same SRA.

8.2.3 Index Sequential Access Method (ISAM)

- It is different method of accessing file which is built on the top of sequential access method. These methods construct an index for the file. It controls the pointer by using index.
- The index like an index in the book that contains the pointer to the various blocks. To search a record in the file, we first search the index and then by the help of pointer we access the file directly.
- In this method, an index value is generated for each primary key and mapped with the record. This index contains the address of the record in the file.
- If any record has to be retrieved based on its index value, then the address of the data block is fetched and the record is retrieved from the memory.

Advantages of ISAM:

1. Records are processed efficiently in both sequential and random order.
2. Data access is direct and fast.
3. Centrally maintained data can be kept up-to-date.

Disadvantages of ISAM:

1. As file grows, its performance deteriorates rapidly because of overflows and thus reorganization.
2. ISAM lowers the system's efficiency.

8.3 FILE STRUCTURE

- Files can be structured in any of the several ways. The three common possibilities are described in Fig. 8.1.

1. **Stream of Bytes (See Fig. 8.1 (a))**
   - OS (Operating System) considers a file to be unstructured.
   - Simplifies file management for the OS. Applications can impose their own structure.
   - Used by UNIX, Windows and most modern Operating Systems.
2. **Records (See Fig. 8.1 (b))**
   - A file is a sequence of fixed length record, each with some internal structure.
   - Collection of bytes is treated as a unit.
   - For example: employee record.
   - Operations are at the level of records (read_rec, write_rec).
   - File is a collection of similar records. OS can optimize operations on records.
3. **Tree of Records (see Fig. 8.1 (c))**
   - A file consists of a tree of records, not necessarily all the same length.
   - Records can be of variable length.
   - Each record has an associated key. The record retrieval is based on the key.

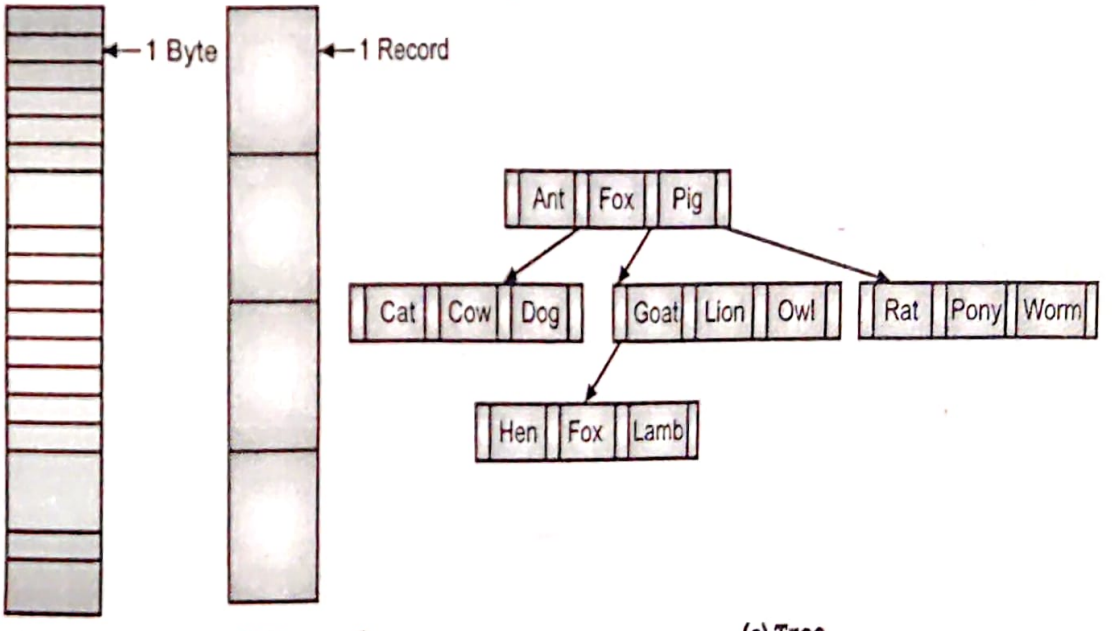


Fig. 8.1: Various File structures

8.3.1 Directory Structure [W-18]

- File systems usually consist of files separated into groups called directories (also called folders). Directories can contain files or additional directories.

Operations performed on directory:

- When we are considering a particular directory structure, the operations that are performed on a directory are:
  - **Search for a file:** We need to be able to search a directory structure to find the entry for particular file.
  - **Create a file:** New files need to be created and added to directory.
  - **Delete a file:** When a file is deleted, an entry must be removed from directory.
  - **List directory:** We need to be able to list the files in directory and contents of the directory entry for each file in the list.
  - **Update directory:** Because some file attributes are stored in the directory a change in one of these attributes requires a change in the corresponding directory entry.

Types of directory structure:

- The directory can be viewed as a symbol table translates file names into their directory entries. If we take such a view, we see that the directory itself can be organized in many ways.

1. Single level directory:

- The simplest directory structure is the single level directory. All files are contained in a same directory, which is easy to support and understand.

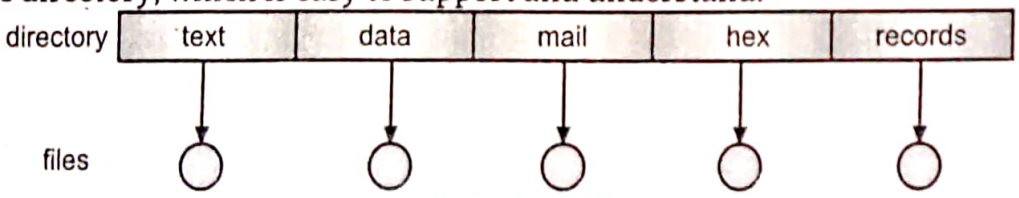


Fig. 8.2: Single level directory

- A single-level directory has significant limitation, however, when 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.
- A single-level directory have problem of filenames using by different users. The solution is to create separate directory for each user.

2. Two-level directory:

- In the two-level directory structure, each user has own User File Directory (UFD) when user log in, the system's Master File Directory (MFD) is searched. The MFD is indexed by user name or account number and each entry points to the UFD for that user.

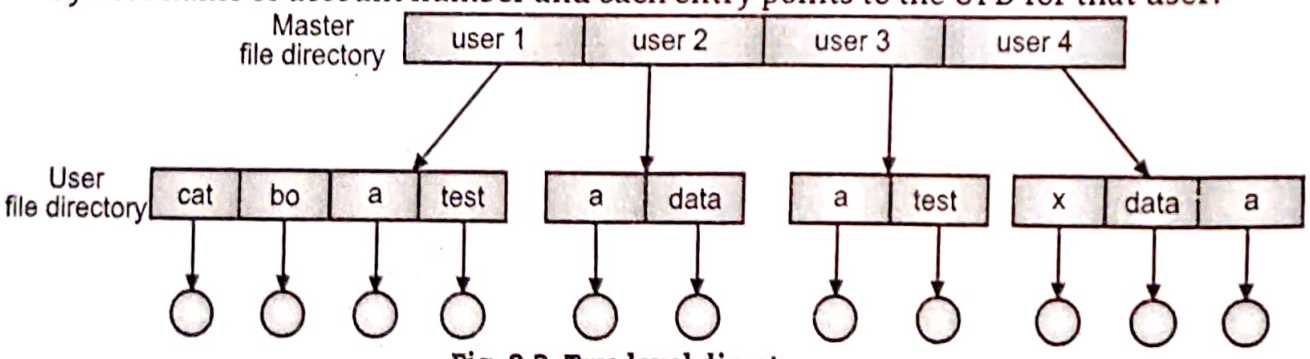


Fig. 8.3: Two level directory

- To create file for a user, the operating system searches only that user's UFD to ascertain whether another file of that name exists. To delete file, the operating system continues its search to the local UFD. The user directories themselves must be created and deleted as necessary.
- The two-level directory solves the name-collision problem, it still has disadvantage. This structure isolates one user from another. This isolation is an advantage when the user is completely independent but is disadvantage when the user wants to co-operate on same task and to access one another's file.
- The two-level hierarchy eliminates name conflicts among users but is not satisfactory for user with large number of files. We have to extend the two-level tree into the directory structure to tree of arbitrary height.

3. Tree-structured directory:

- It is an inverted tree structure with a single root. The topmost directory in the hierarchy is called the root directory. A directory is called the parent directory of the subdirectories and files in it. It can contain any number of directories. A directory can contain any number of files and subdirectories.
- The MS-DOS system is structured as a tree. The tree has a root directory. Every file in the system has a unique path name.
- A path name is the path from the root, through all the subdirectories to specified file.

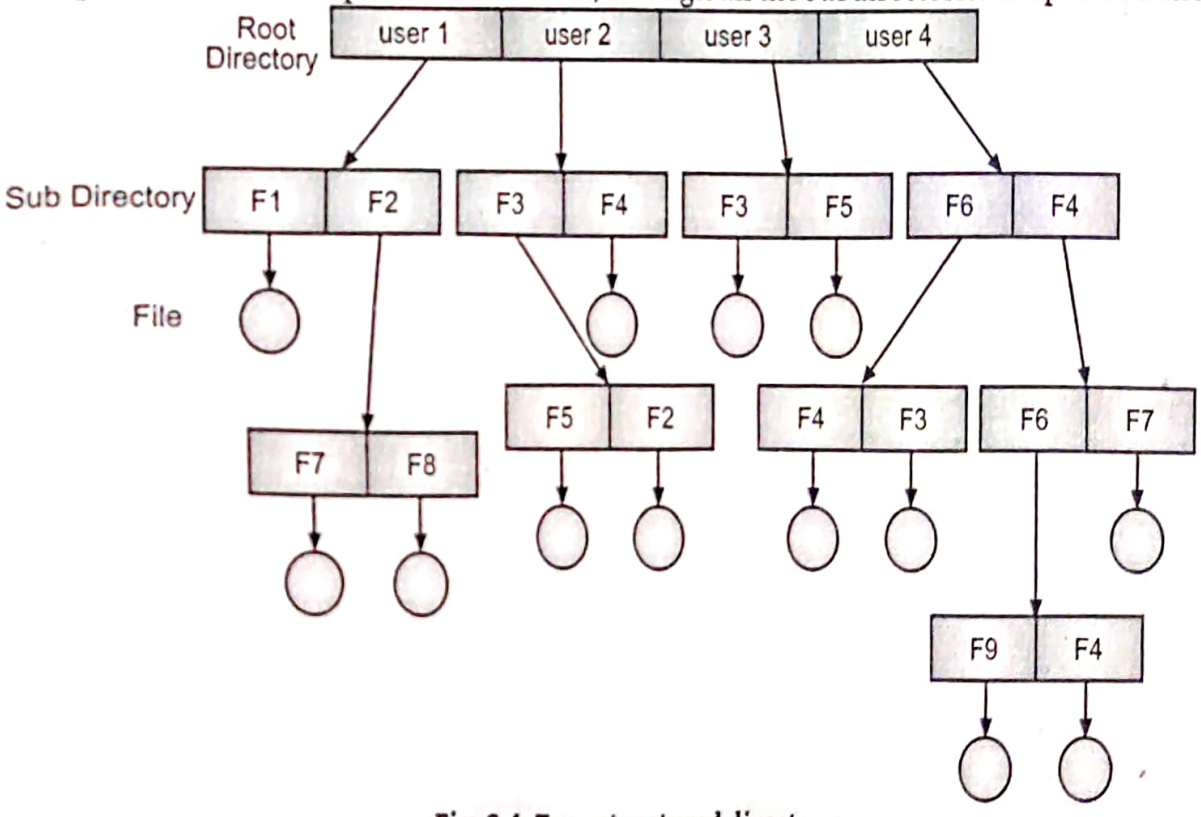


Fig. 8.4: Tree-structured directory

4. Acyclic graph Directory:

- Consider two programmers who are working on a joint project. The files associates with that project can be stored in a subdirectory, separating them from other projects and files of the two programmers.
- But since both the programmers are equally responsible for the project both want the subdirectory to be in their own directories. The common subdirectory should be shared.
- A shared directory or file will exist in two different places at once.

- A tree structure prohibits the sharing of files or directories. But an acyclic graph allows directories to have shared subdirectories of files, (Fig. 8.5).

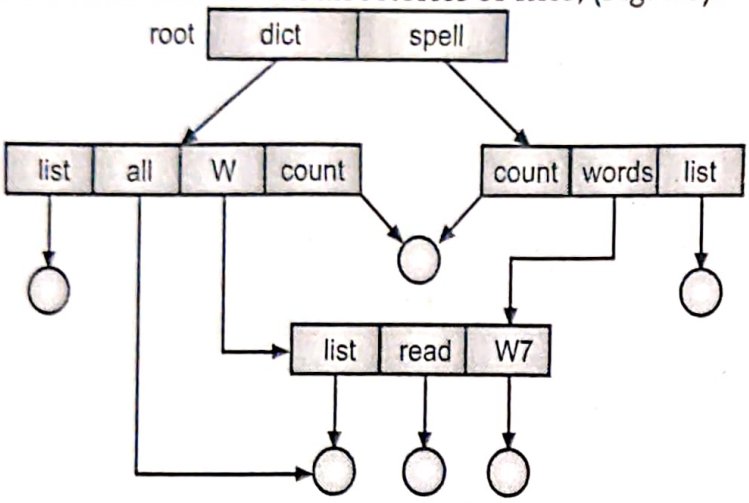


Fig. 8.5: Acyclic Graph Directory

- The same file subdirectory may be in two different directories. An acyclic graph, that is, a graph with no cycles is a natural generation of tree structured directory schemes.

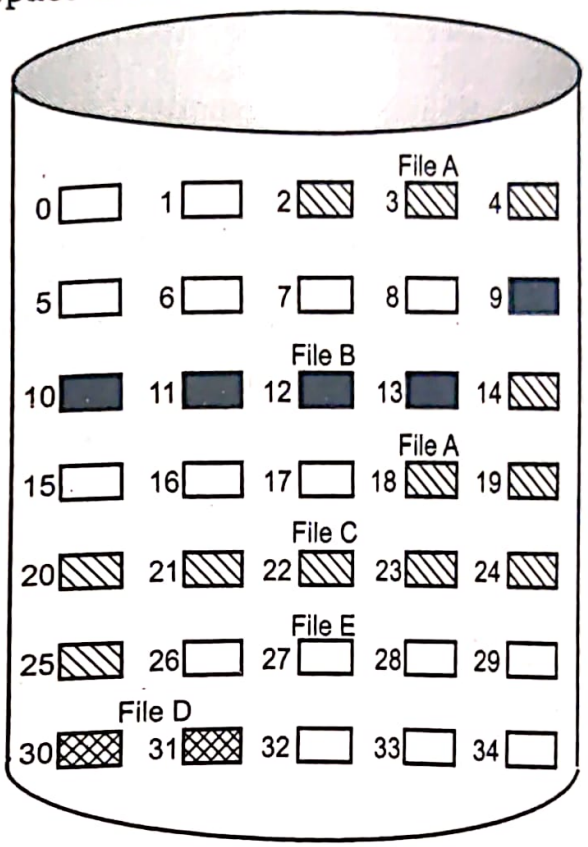
8.4 ALLOCATION METHODS

- The file allocation methods define how the files are stored in the disk blocks.
- Several issues are involved in file allocation:
  1. When a new file is created, is the maximum space required for the file allocated at once?
  2. Space is allocated to file as one or more contiguous units. What size of portion should be used for file allocation?
  3. What sort of data structure is used to keep track of the blocks to a file? Such a table is typically referred to as File Allocation Table (FAT).
- A static allocation requires that the maximum size of file be declared at the time of file creation request. In number of cases the value can be reliably estimate, but many cases, it is difficult.
- In those cases users and application programmers would tend to overestimate file size. This is clearly wasteful from the point view of secondary storage allocation.
- Thus, there are advantages to the use of dynamic allocation, which allocates space to a file in blocks are needed.
- There are different types of file allocation methods, but we mainly use three types of file allocation methods.
  - Contiguous allocation
  - Linked list allocation
  - Indexed allocation
- These methods provide quick access to the file blocks and also the utilization of the disk space in an efficient manner.

8.4.1 Contiguous Allocation [S-18]

- Contiguous allocation is one of the most used methods for allocation.
- With this method a single contiguous set of blocks is allocated to a file creation (Fig. 8.6). Thus, this is the static allocation method, using variable-size portion.
- The FAT needs just a one entry. For each file, showing starting block and the length of file. Contiguous allocation is best for sequential file.

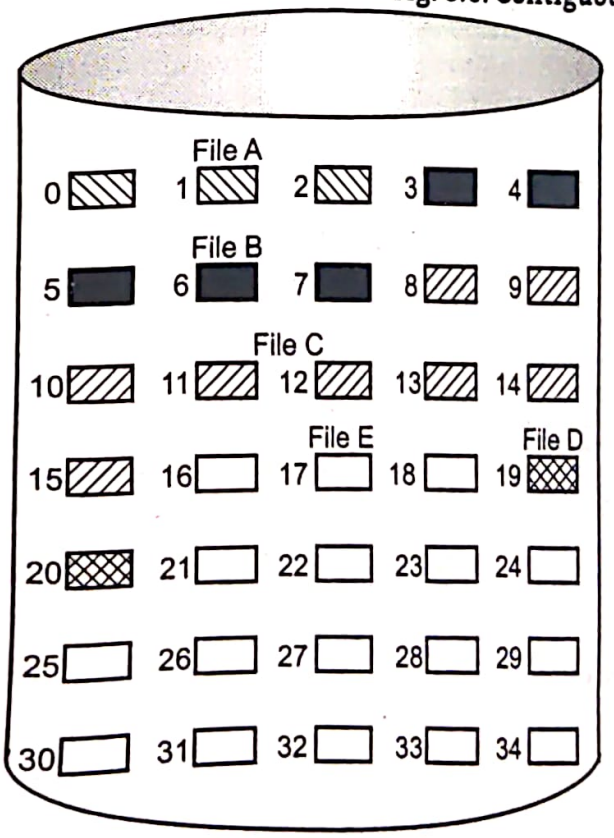
- Contiguous allocation has some problems. External fragmentation will occur, making it difficult to find contiguous blocks of space of sufficient length.
- For time to time, it is necessary to do a compaction algorithm to free up additional space of disk.



File Allocation Table

File Name	Start Block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

Fig. 8.6: Contiguous File Allocation



File Allocation Table

File Name	Start Block	Length
File A	0	3
File B	3	5
File C	8	8
File D	19	2
File E	16	3

Fig. 8.7: Contiguous File Allocation (After Compaction)

Advantages:

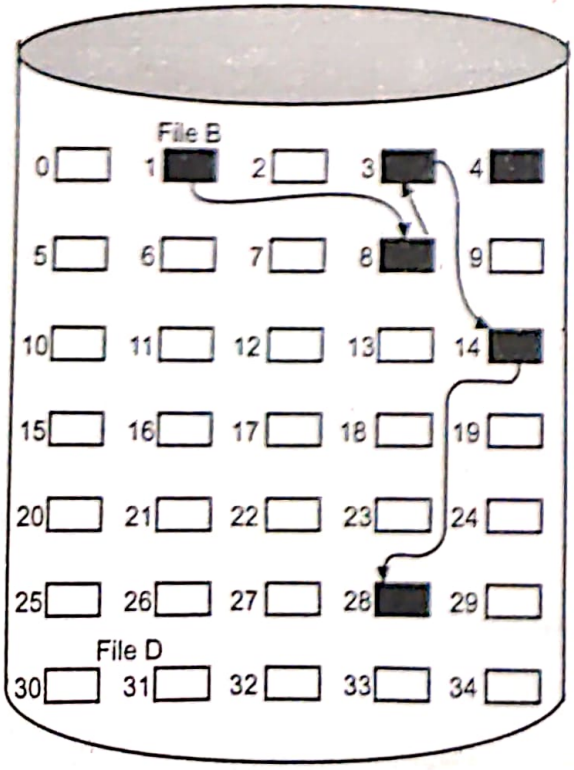
1. Supports both sequential and direct access methods.
2. Contiguous allocation is the best form of allocation for **sequential files**. Multiple blocks can be brought in at a time to improve I/O performance for sequential processing.
3. It is also easy to retrieve a single block from a file. For example, 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.

Disadvantages:

1. Suffers from external fragmentation.
2. Very difficult to find contiguous blocks of space.
3. Also with per-allocation, it is necessary to declare the size of the file at the time of creation which many times is difficult to estimate.

8.4.2 Linked Allocation [S-19, 23; W-22]

- Linked allocation solves all the problem of contiguous allocation (Fig. 8.8). With chained allocation, each file is a linked list of disk blocks; the blocks may be scattered anywhere on the disk.
- Again, the file allocation table needs just a single entry for each file, showing starting block and length of file.
- Although static allocation is possible, it is more common simply to allocate block as needed. There is no external fragmentation because only one block at a time is needed.
- The major problem is that it can be used efficiently only for sequential file. To find its block, we must start at the beginning of that file and follow the pointer until we get to the i^{th} block.
- Linked allocation is inefficient to support a direct access files. Another disadvantage to chained allocation is the space required for the pointers.



File Allocation Table

File Name	Start Block	Length
...	...	...
File B	1	5
...	...	...

Fig. 8.8: Linked/Chained Allocation

Advantages:

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.

Disadvantages:

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.

8.4.3 Indexed Allocation

W-18, 22; S-22

- Linked allocation solves the external fragmentation and size declaration problem and contiguous allocation.
- However, chained allocation cannot support efficient direct access, since pointers are scattered with the blocks themselves all over the disk and need to be retrieved in order.
- Indexed allocation solves this problem by bringing all the pointer is together into one location: the index block. In this case, the FAT contains a separate one-level index for each file, the index has one entry for each portion allocated to file.
- File indexes are not physically stored as part of the FAT, but it is kept in a separate block and entry for the file in the FAT points to that block.
- Allocation may be on the basis of either fixed-sized blocks or variable-size partitions. Allocation by blocks eliminates external fragmentation, whereas allocation by variable size portions improve locality.
- Indexed allocation supports both sequential and direct access to the file and thus is the most popular form of file allocation.

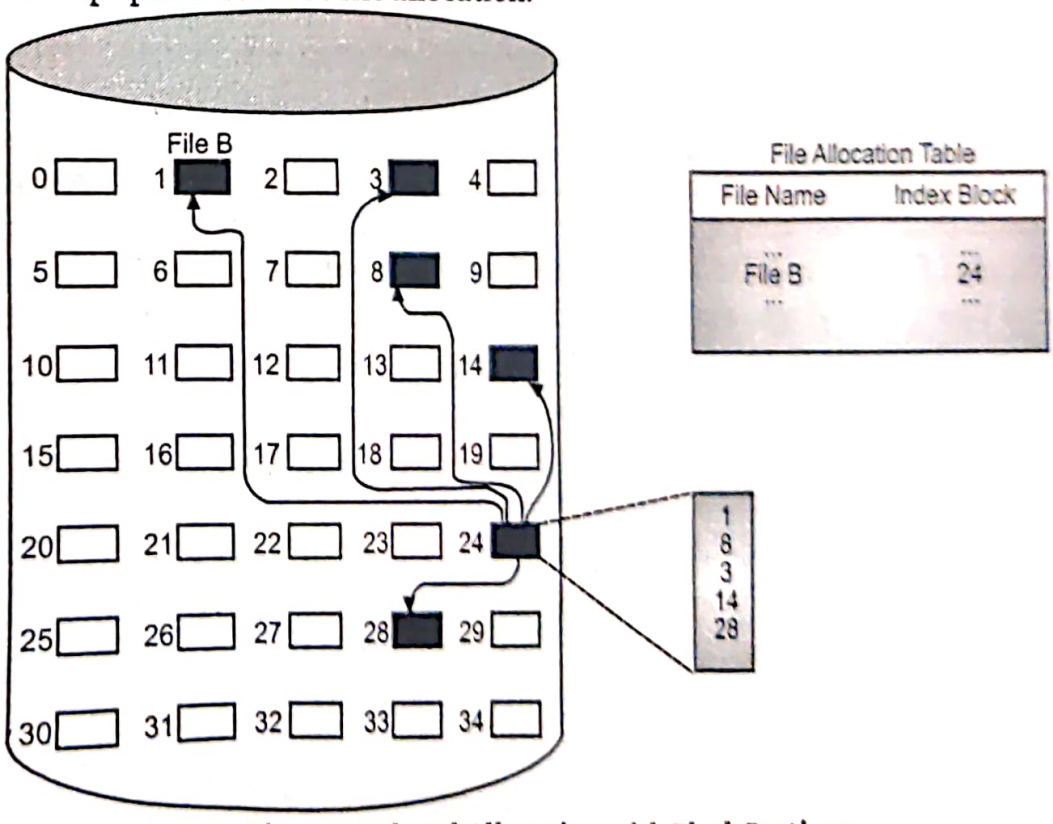


Fig. 8.9: Indexed Allocation with Block Portions

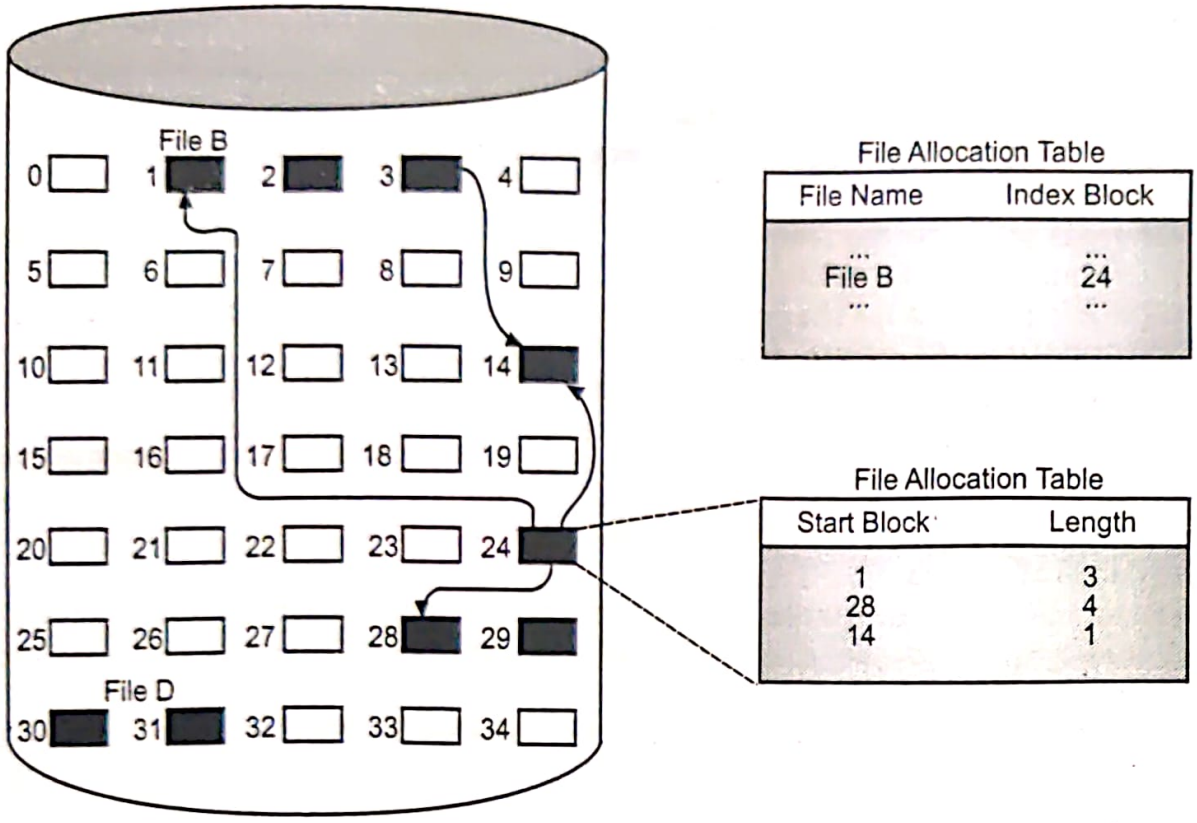


Fig. 8.10: Indexed Allocations with Variable-Length Portions

Advantages:

1. Does not suffer from external fragmentation.
2. Support both sequential and direct access to the file.

Disadvantages:

1. In index allocation, pointer overhead is more.
2. We can lose the entire file if an index block is not correct.
3. It is totally wastage to create an index for a small file.

8.5 FREE SPACE MANAGEMENT [W-18; S-22]

- As we know that the memory space in the disk is limited. So we need to use the space of the deleted files for the allocation of the new file. One optical disk allows only one write at a time in the given sector and thus it is not physically possible to reuse it for other files.
- The system maintains a free space list by keep track of the free disk space.
- The free space list contains all the records of the free space disk block. The free blocks are those which are not allocated to other file or directory.
- When we create a file we first search for the free space in the memory and then check in the free space list for the required amount of space that we require for our file. If the free space is available then allocate this space to the new file.
- After that, the allocating space is deleted from the free space list. Whenever we delete a file then its free memory space is added to the free space list.
- The process of looking after and managing the free blocks of the disk is called Free Space Management.

- There are some methods or techniques to implement a free space list. These are as follows:
  1. Bitmap
  2. Linked list
  3. Grouping
  4. Counting

8.5.1 Bit Vector (Bitmap)

- This technique is used to implement the free space management. When the free space is implemented as the bitmap or bit vector then each block of the disk is represented by a bit. When the block is free its bit is set to 1 and when the block is allocated the bit is set to 0.
- The main advantage of the bitmap is it is relatively simple and efficient in finding the first free block and also the consecutive free block in the disk. Many computers provide the bit manipulation instruction which is used by the users.
- The calculation of the block number is done by the formula:

  (Number of bits per words) × (Number of 0 value word) + Offset of first 1-bit
- Apple Macintosh operating system uses the bitmap method to allocate the disk space.

Example 1: Assume the following blocks are free. Rest of the blocks are allocated:

2,3,4,5,9,10,13

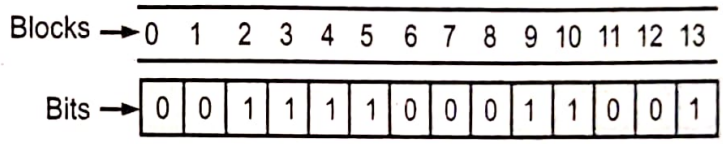


Fig. 8.11: Bit Vector Technique

Example 2: Consider a disk where blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26, and 27 are free and the rest of the blocks are allocated. The free-space bitmap would be: 00111100111111000110000001110000.

Advantages:

1. This technique is relatively simple.
2. This technique is very efficient to find the free space on the disk.

Disadvantages:

1. This technique requires a special hardware support to find the first 1 in a word it is not 0.
2. This technique is not useful for the larger disks.

8.5.2 Linked list

- This is another technique for free space management. In this linked list, the entire free block is maintained. In this, there is a *head pointer* which points the first free block of the list which is kept in a special location on the disk.
- This block contains the pointer to the next block and the next block contains the pointer of another next and this process is repeated. By using this list it is not easy to search the free list.

- This technique is not sufficient to traverse the list because we have to read each disk block that requires I/O time. So traversing in the free list is not a frequent action.

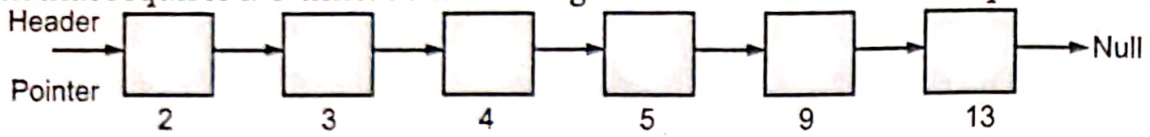


Fig 8.12: Example of Linked List

- In this example, we see that keep block 2 is the first free block which points to another block which contains the pointer of the 3 blocks and 3 blocks contain the pointer to the 4 blocks and this contains the pointer to the 5 block then 5 block contains the pointer to the next block and this process is repeated at the last .

Advantages:

1. Whenever a file is to be allocated a free block, the operating system can simply allocate the first block in free space list and move the head pointer to the next free block in the list.

Disadvantages:

1. Searching the free space list will be very time consuming; each block will have to be read from the disk, which is read very slowly as compared to the main memory.
2. Not Efficient for faster access.

8.5.3 Grouping

- This is also the technique of free space management. In this, there is a modification of the free-list approach which stores the address of the n free blocks. In this the first n-1 blocks are free but the last block contains the address of the n blocks.
- When we use the standard linked list approach the addresses of a large number of blocks can be found very quickly. In this approach, we cannot keep a list of n free disk addresses but we keep the address of the first free block.

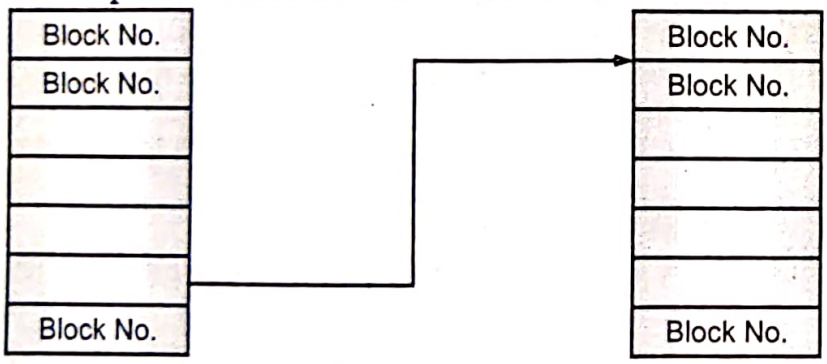


Fig. 8.13: Grouping Technique

8.5.4 Counting

- Counting is another approach for free space management. Generally, some contiguous blocks are allocated but some are free simultaneously.
- The free space is allocated to a process according to the contiguous allocation algorithm or clustering.
- So, we cannot keep the list of n free block address but we can keep the address of the first free block and then the numbers of n free contiguous block which follows the first block.

- There is an entry in the free space list that consist the address of the disk and a count variable. This method of free space management is similar to the method of allocating blocks.
- We can store these entries in the B-tree in place of the linked list. So the operations like lookup, deletion, insertion are efficient.

Summary

- A file is a named collection of related information that is recorded on a secondary storage. Commonly, files represent programs (source and object forms) and data.
- The file may have attributes like name, creator, date, type, permissions etc.
- There are various operations which can be implemented on a file such as create, write, delete, truncate, read, re-position etc.
- There are three ways to access a file into a computer system: Sequential-Access, Direct Access, Index sequential Method.
- In Sequential-Access, information in the file is processed in order, one record after the other.
- Direct access is very inefficient with linked allocation. Indexed allocation may require substantial overhead for its index block.
- File allocation methods define how the files are stored in the disk blocks.
- There are three main file allocation methods: Contiguous Allocation, Linked Allocation, Indexed Allocation.
- Contiguous allocation can suffer from external fragmentation.
- Free-space allocation methods also influence the efficiency of disk-space use, the performance of the file system, and the reliability of secondary storage.
- The methods used include bit vectors and linked lists. Optimizations include grouping, counting, and the FAT, which places the linked list in one contiguous area.

Check Your Understanding

1. _____ is a unique tag usually a number identifies the file within the file system.
   (a) File identifier (b) File name
   (c) File type (d) None of the mentioned
2. To create a file _____.
   (a) allocate the space in file system
   (b) make an entry for new file in directory
   (c) allocate the space in file system & make an entry for new file in directory
   (d) none of the mentioned
3. By using the specific system call, we can _____
   (a) open the file (b) read the file
   (c) write into the file (d) all of the mentioned

4. File type can be represented by _____
   (a) file name (b) file extension
   (c) file identifier (d) none of the mentioned
5. Which file is a sequence of bytes organized into blocks understandable by the system's linker?
   (a) object file (b) source file
   (c) executable file (d) text file
6. What is the mounting of file system?
   (a) creating of a filesystem.
   (b) deleting a filesystem.
   (c) attaching portion of the file system into a directory structure.
   (d) removing the portion of the file system into a directory structure.
7. Which one of the following explains the sequential file access method?
   (a) random access according to the given byte number.
   (b) read bytes one at a time, in order.
   (c) read/write sequentially by record.
   (d) read/write randomly by record.
8. When will file system fragmentation occur?
   (a) unused space or single file are not contiguous.
   (b) used space is not contiguous.
   (c) unused space is non-contiguous.
   (d) multiple files are non-contiguous.
9. The larger the block size, the _____ the internal fragmentation.
   (a) greater (b) lesser
   (c) same (d) none of the mentioned
10. In the sequential access method, information in the file is processed _____.
    (a) one disk after the other, record access does not matter.
    (b) one record after the other.
    (c) one text document after the other.
    (d) none of the mentioned.
11. Sequential access method _____ on random access devices.
    (a) works well
    (b) does not work well
    (c) maybe works well and does not work well
    (d) none of the mentioned
12. The direct access method is based on a _____ model of a file, as _____ allow random access to any file block.
    (a) magnetic tape, magnetic tapes (b) tape, tapes
    (c) disk, disks (d) all of the mentioned

13. For a direct access file _____.
    (a) there are restrictions on the order of reading and writing.
    (b) there are no restrictions on the order of reading and writing.
    (c) access is restricted permission wise.
    (d) access is not restricted permission wise.
14. A relative block number is an index relative to _____.
    (a) the beginning of the file (b) the end of the file
    (c) the last written position in file (d) none of the mentioned
15. The index contains _____.
    (a) names of all contents of file (b) pointers to each page
    (c) pointers to the various blocks (d) all of the mentioned
16. For large files, when the index itself becomes too large to be kept in memory?
    (a) index is called
    (b) an index is created for the index file
    (c) secondary index files are created
    (d) all of the mentioned

Answers

1. (a)	2. (c)	3. (d)	4. (b)	5. (a)	6. (c)	7. (b)	8. (a)	9. (a)	10. (b)
11. (a)	12. (c)	13. (b)	14. (a)	15. (c)	16. (b)				

Practice Questions

Q.I Answer the following questions in short:

1. Define the term file.
2. State various attributes of files.
3. What are the types of files?
4. What is meant by file allocation?
5. List basic operations on file.
6. What are the different types of access methods?
7. What is counting technique?

Q.II Answer the following questions:

1. With suitable diagram describe direct access method.
2. Describe the term Indexed allocation in detail.
3. With the help of diagram describe file structure.
4. With the help of diagram describe sequential access method.
5. What is meant by free space management?
6. Write a short note on Linked Allocation in file.
7. Compare sequential access and direct access.

8. Explain Direct Access method with advantages and disadvantages.
9. Explain contiguous Allocation method in detail.
10. List and explain any two operations that can be performed on file.
11. List and explain different attributes related to file.

Q.III Define terms:

1. Bit vector
2. Linked list
3. Grouping
4. File attributes
5. Directory

Previous Exam Questions

Summer 2018

1. Explain Direct Access method with advantages and disadvantages. **[4 M]**

Ans. Refer to Section 8.2.2.

2. Explain contiguous memory allocation method in detail. **[4 M]**

Ans. Refer to Section 8.4.1

Winter 2018

1. List various operations on files. **[2 M]**

Ans. Refer to Section 8.1.2.

2. Write a short note on File Directories. **[4 M]**

Ans. Refer to Section 8.3.1.

3. Explain Indexed Allocation briefly. **[4 M]**

Ans. Refer to Section 8.4.3.

4. What is meant by free space management? Define Bit vector and Grouping. **[4 M]**

Ans. Refer to Section 8.5.

Summer 2019

1. List any four attributes on files. **[2 M]**

Ans. Refer to Section 8.1.1.

2. Explain advantages and disadvantages of Linked allocation Method. **[4 M]**

Ans. Refer to Section 8.4.2.

■■■