

File Management

A file is a collection of similar records.

- The data can't be written on to the secondary storage unless they are within a file.
- Files represent both the program and the data. Data can be numeric, alphanumeric, alphabetic or binary.
- Many different types of information can be stored on a file ---Source program, object programs, executable programs, numeric data, payroll recorder, graphic images, sound recordings and so on.

File Attributes

File attributes varies from one OS to other. The common file attributes are:

1. **Name:-** The symbolic file name is the only information kept in human readable form.
2. **Identifier:-** The unique tag, usually a number, identifies the file within the file system. It is the non-readable name for a file.
3. **Type:-** This information is needed for those systems that supports different types.
4. **Location:-** This information is a pointer to a device and to the location of the file on that device.
5. **Size:-** The current size of the file and possibly the maximum allowed size are included in this attribute.
6. **Protection:-** Access control information determines who can do reading, writing, execute and so on.
7. **Time, data and User Identification:-** This information must be kept for creation, last modification and last use. These data are useful for protection, security and usage monitoring.

File types:

A file has a certain defined structures according to its type:-

1. **Text file:-** Text file is a sequence of characters organized in to lines.
2. **Object file:-** Object file is a sequence of bytes organized in to blocks understandable by the systems linker.

3. **Executable file:-** Executable file is a series of code section that the loader can bring in to memory and execute.
4. **Source File:-** Source file is a sequence of subroutine and function, each of which are further organized as declaration followed by executable statements.

File operation:

File is an abstract data type. To define a file we need to consider the operation that can be performed on the file.

Basic operations of files are:-

1. **Creating a file:-** Two steps are necessary to create a file. First space in the file system for file is found. 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:-** System call is mainly used for writing in to the file. System call specify the name of the file and the information i.e., to be written on to the file. Given the name the system search the entire directory for the file. The system must keep a write pointer to the location in the file where the next write to be taken place.
3. **Reading a file:-** To read a file system call is used. It requires the name of the file and the memory address. Again the directory is searched for the associated directory and system must maintain a read pointer to the location in the file where next read is to take place.
4. **Delete a file:-** System will search for the directory for which file to be deleted. If entry is found it releases all free space. That free space can be reused by another file.
5. **Truncating the file:-** User may want to erase the contents of the file but keep its attributes. Rather than forcing the user to delete a file and then recreate it, truncation allows all attributes to remain unchanged except for file length.
6. **Repositioning within a file:-** The directory is searched for appropriate entry and the current file position is set to a given value. Repositioning within a file does not need to involve actual i/o. The file operation is also known as file seeks.

In addition to this basis 6 operations the other two operations include appending new information to the end of the file and renaming the existing file. These primitives can be combined to perform other two operations.

Most of the file operation involves searching the entire directory for the entry associated with the file. To avoid this OS keeps a small table containing information about an open file (the open table). When a file operation is requested, the file is specified via index in to this table. So searching is not required.

Several piece of information are associated with an open file:-

- ✓ **File pointer:-** on systems that does not include offset an a part of the read and write system calls, the system must track the last read-write location as current file position pointer. This pointer is unique to each process operating on a file.
- ✓ **File open count:-** As the files are closed, the OS must reuse its open file table entries, or it could run out of space in the table. Because multiple processes may open a file, the system must wait for the last file to close before removing the open file table entry. The counter tracks the number of copies of open and closes and reaches zero to last close.
- ✓ **Disk location of the file:-** The information needed to locate the file on the disk is kept in memory to avoid having to read it from the disk for each operation.
- ✓ **Access rights:-** Each process opens a file in an access mode. This information is stored on per-process table the OS can allow OS deny subsequent I/O request.

Access Methods:

The information in the file can be accessed in several ways. Different file access methods are:-

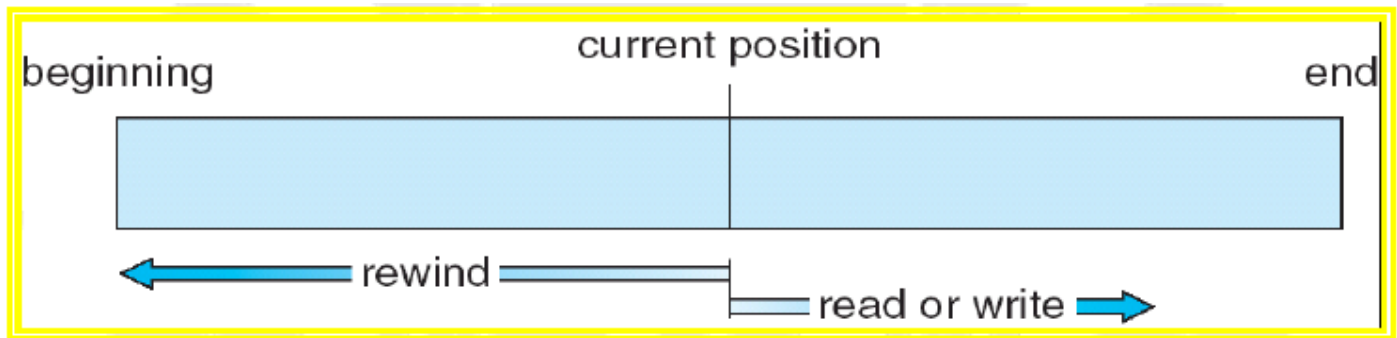
1. Sequential Access:-

Sequential access is the simplest access method. Information in the file is processed in order, one record after another. Editors and compilers access the files in this fashion.

Normally read and write operations are done on the files. A read operation reads the next portion of the file and automatically advances a file pointer, which track next i/I track.

Write operation appends to the end of the file and such a file can be next to the beginning.

Sequential access depends on a tape model of a file.



2. Direct Access OR relative access

- ✓ Direct access allows random access to any file block.
- ✓ This method is based on disk model of a file.
- ✓ A file is made up of fixed length logical records. It allows the program to read and write records rapidly in any order.
- ✓ A direct access file allows arbitrary blocks to be read or written.

Example:-User may need block 13, then read block 99 then write block 12.

For searching the records in large amount of information with immediate result, the direct access method is suitable. Not all OS support sequential and direct access. Few OS use sequential access and some OS use direct access. It is easy to simulate sequential access on a direct access but the reverse is extremely inefficient.

Direct access is based on disk model.

Indexing Method:-

- The index is like an index at the end of a book which contains pointers to various blocks.
- To find a record in a file, we search the index and then use the pointer to access the file directly and to find the desired record.

With large files index file itself can be very large to be kept in memory. One solution is to create an index to the index files itself. The primary index file

would contain pointer to secondary index files which would point to the actual data items.

Two types of indexes can be used:-

- **Exhaustive index:-** Contain one entry for each of record in the main file. An index itself is organized as a sequential file.
- **Partial index:-** Contains entries to records where the field of interest exists with records of variable length, some record will not contain an fields. When a new record is added to the main file, all index files must be updated.

Directory structure:

File System structure:

Allocation methods:

Free-space management and Directory implementation:

Structure of Linux Operating System:

Exploring the Directory Structure:

Naming Files and Directories:

Concept of shell:

Types of shell:

Editors for shell programming (e.g. vi):

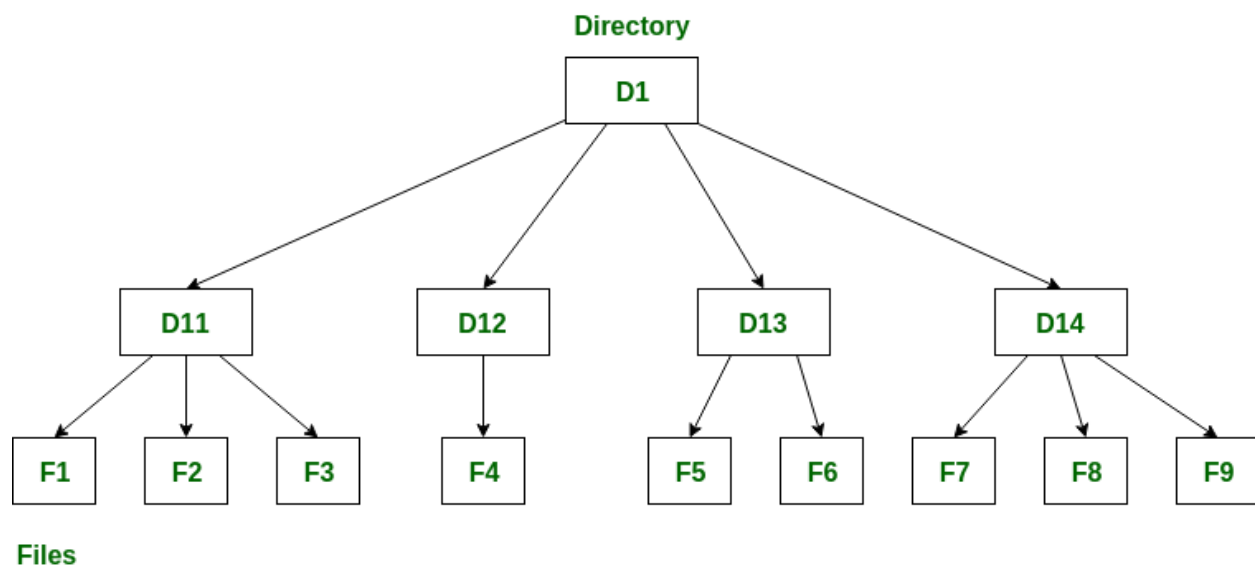
Basics of Shell programming:

Directory overview

Structures of Directory in Operating System

A directory is a container that is used to contain folders and files. It organizes files and folders in a hierarchical manner. In other words, directories are like folders that help organize files on a computer. Just like you use folders to keep your papers and documents in order, the operating system uses directories to keep track of files and where they are stored. Different structures of directories can be used to organize these files, making it easier to find and manage them.

Understanding these directory structures is important because it helps in efficiently organizing and accessing files on your computer. Following are the logical structures of a directory, each providing a solution to the problem faced in the previous type of directory structure. Structures of Directory in Operating System.



Different Types of Directory in OS

In an operating system, there are different types of directory structures that help organize and manage files efficiently.

Directories in an OS can be single-level, two-level, or hierarchical. To gain a comprehensive understanding of file systems, explore the GATE CS Self-Paced Course which covers operating systems in great detail.

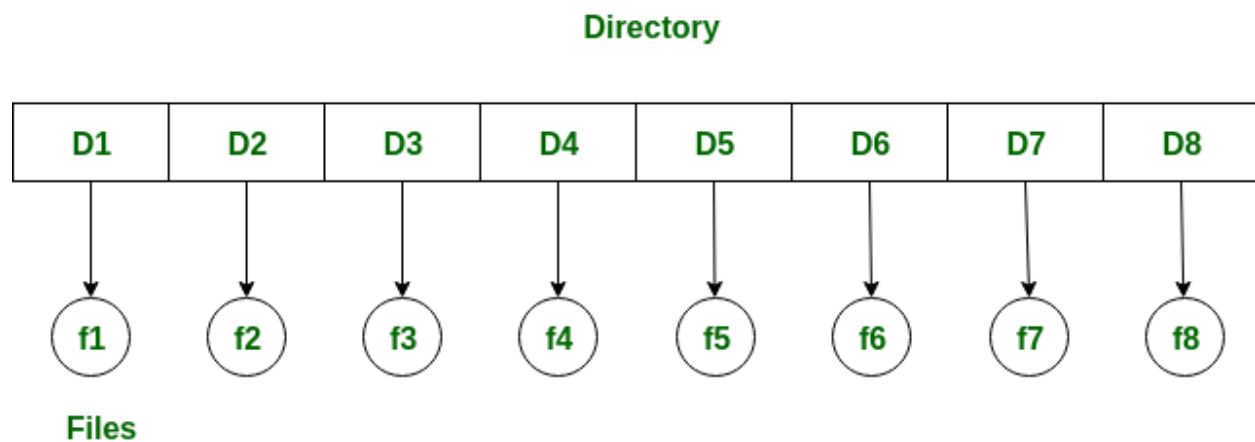
Each type of directory has its own way of arranging files and directories, offering unique benefits and features. These are:

1. **Single-Level Directory**
2. **Two-Level Directory**
3. **Tree Structure/ Hierarchical Structure**
4. **Acyclic Graph Structure**
5. **General-Graph Directory Structure**

Single-Level Directory

The single-level directory is the simplest directory structure. In it, all files are contained in the same directory which makes it easy to support and understand.

A single level directory has a significant limitation, however, when the number of files increases or when the system has more than one user. Since all the files are in the same directory, they must have a unique name. If two users call their dataset test, then the unique name rule is violated.



Advantages

- Since it is a single directory, so its implementation is very easy.
- If the files are smaller in size, searching will become faster.
- The operations like file creation, searching, deletion, updating are very easy in such a directory structure.
- Logical Organization : Directory structures help to logically organize files and directories in a hierarchical structure. This provides an easy way to navigate and manage files, making it easier for users to access the data they need.

- **Increased Efficiency:** Directory structures can increase the efficiency of the file system by reducing the time required to search for files. This is because directory structures are optimized for fast file access, allowing users to quickly locate the file they need.
- **Improved Security :** Directory structures can provide better security for files by allowing access to be restricted at the directory level. This helps to prevent unauthorized access to sensitive data and ensures that important files are protected.
- **Facilitates Backup and Recovery :** Directory structures make it easier to backup and recover files in the event of a system failure or data loss. By storing related files in the same directory, it is easier to locate and backup all the files that need to be protected.
- **Scalability:** Directory structures are scalable, making it easy to add new directories and files as needed. This helps to accommodate growth in the system and makes it easier to manage large amounts of data.

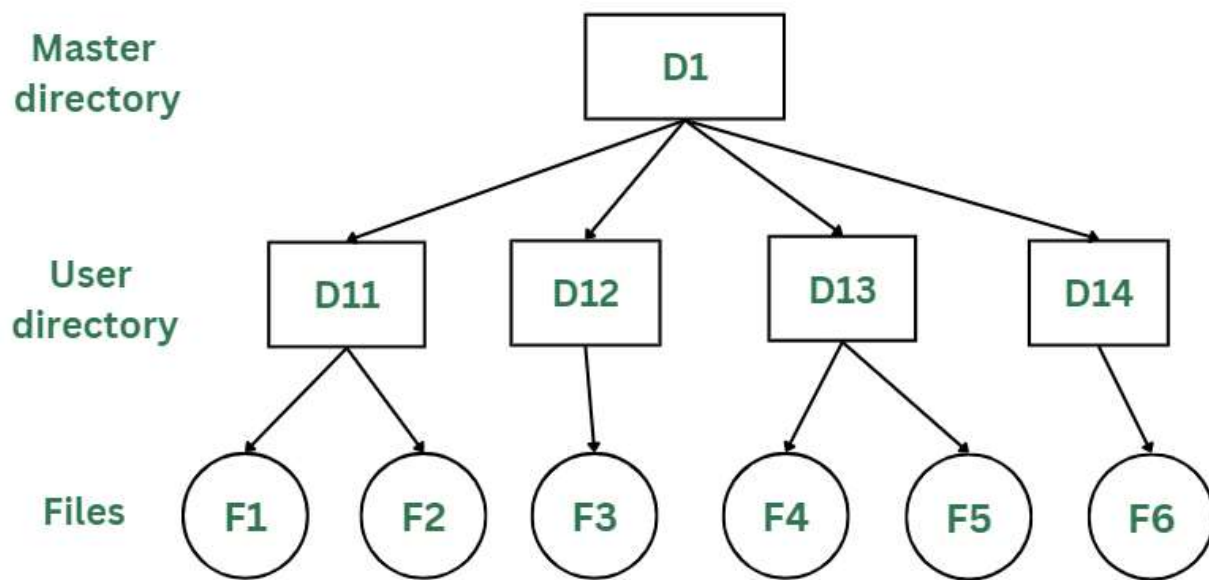
Disadvantages

- There may chance of name collision because two files can have the same name.
- Searching will become time taking if the directory is large.
- This can not group the same type of files together.

Two-Level Directory

As we have seen, a single level directory often leads to confusion of files names among different users. The solution to this problem is to create a separate directory for each user.

In the two-level directory structure, each user has their own user files directory (UFD). The UFDs have similar structures, but each lists only the files of a single user. System's master file directory (MFD) is searched whenever a new user id is created.



Advantages

- The main advantage is there can be more than two files with same name, and would be very helpful if there are multiple users.
- A security would be there which would prevent user to access other user's files.
- Searching of the files becomes very easy in this directory structure.

Disadvantages

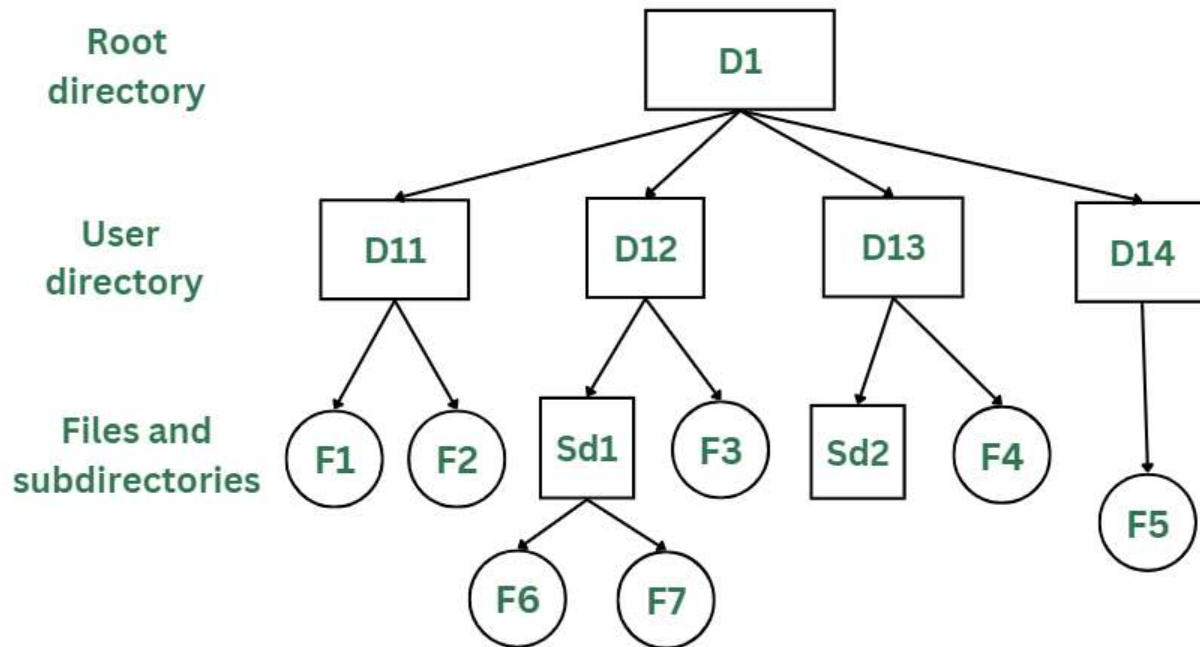
- As there is advantage of security, there is also disadvantage that the user cannot share the file with the other users.
- Unlike the advantage users can create their own files, users don't have the ability to create subdirectories.
- Scalability is not possible because one user can't group the same types of files together.

Tree Structure/ Hierarchical Structure

Tree directory structure of operating system is most commonly used in our personal computers. User can create files and subdirectories too, which was a disadvantage in the previous directory structures.

This directory structure resembles a real tree upside down, where the root directory is at the peak. This root contains all the directories for each user. The users can create subdirectories and even store files in their directory.

A user do not have access to the root directory data and cannot modify it. And, even in this directory the user do not have access to other user's directories. The structure of tree directory is given below which shows how there are files and subdirectories in each user's directory.



Advantages

- This directory structure allows subdirectories inside a directory.
- The searching is easier.
- File sorting of important and unimportant becomes easier.
- This directory is more scalable than the other two directory structures explained.

Disadvantages

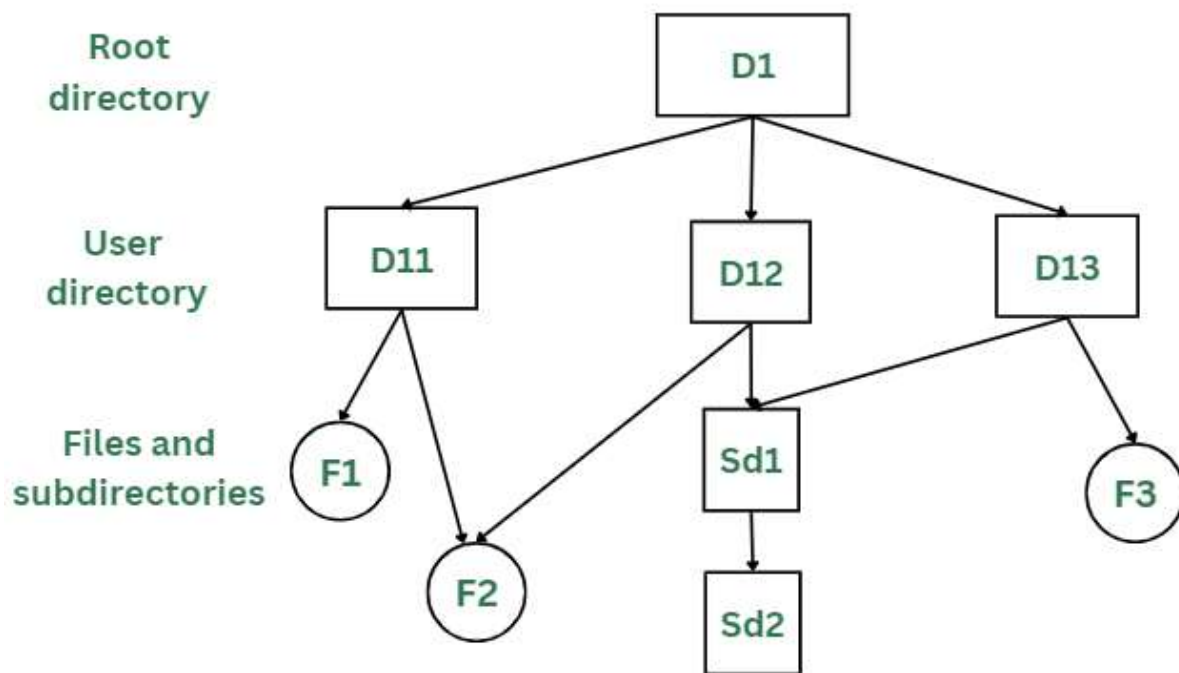
- As the user isn't allowed to access other user's directory, this prevents the file sharing among users.
- As the user has the capability to make subdirectories, if the number of subdirectories increase the searching may become complicated.
- Users cannot modify the root directory data.
- If files do not fit in one, they might have to be fit into other directories.

Acyclic Graph Structure

As we have seen the above three directory structures, where none of them have the capability to access one file from multiple directories. The file or the subdirectory could be accessed through the directory it was present in, but not from the other directory.

This problem is solved in acyclic graph directory structure, where a file in one directory can be accessed from multiple directories. In this way, the files could be shared in between the users. It is designed in a way that multiple directories point to a particular directory or file with the help of links.

In the below figure, this explanation can be nicely observed, where a file is shared between multiple users. If any user makes a change, it would be reflected to both the users.



Advantages

- Sharing of files and directories is allowed between multiple users.
- Searching becomes too easy.
- Flexibility is increased as file sharing and editing access is there for multiple users.

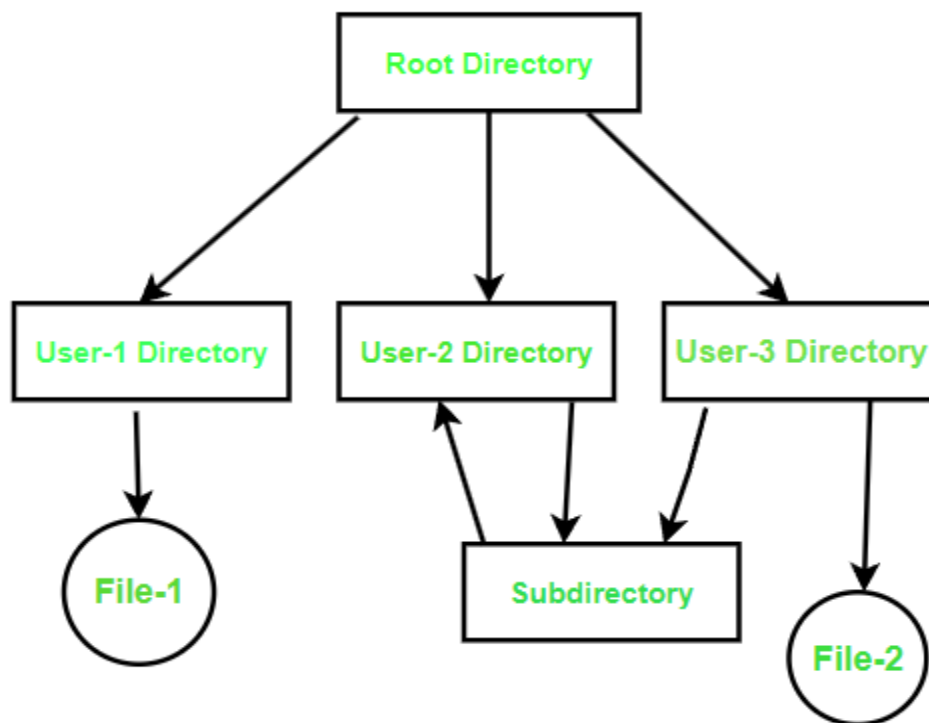
Disadvantages

- Because of the complex structure it has, it is difficult to implement this directory structure.
- The user must be very cautious to edit or even deletion of file as the file is accessed by multiple users.

- If we need to delete the file, then we need to delete all the references of the file in order to delete it permanently.

General-Graph Directory Structure

Unlike the acyclic-graph directory, which avoids loops, the general-graph directory can have cycles, meaning a directory can contain paths that loop back to the starting point. This can make navigating and managing files more complex.



In the above image, you can see that a cycle is formed in the User 2 directory. While this structure offers more flexibility, it is also more complicated to implement.

Advantages of General-Graph Directory

- More flexible than other directory structures.
- Allows cycles, meaning directories can loop back to each other.

Disadvantages of General-Graph Directory

- More expensive to implement compared to other solutions.

- Requires garbage collection to manage and clean up unused files and directories.

Conclusion

Understanding the different directory structures in an operating system is important for efficient file organization and management. Each structure, whether single-level, two-level, tree-structured, acyclic graph, or general graph, offers unique ways to arrange and access files. Choosing the right directory structure helps ensure that files are easy to find, use, and maintain.

File Allocation Methods

The allocation methods define how the files are stored in the disk blocks. There are three main disk space or file allocation methods.

1. Contiguous Allocation
2. Linked Allocation
3. Indexed Allocation

The main idea behind these methods is to provide:

- Efficient disk space utilization.
- Fast access to the file blocks.

All the three methods have their own advantages and disadvantages as discussed below:

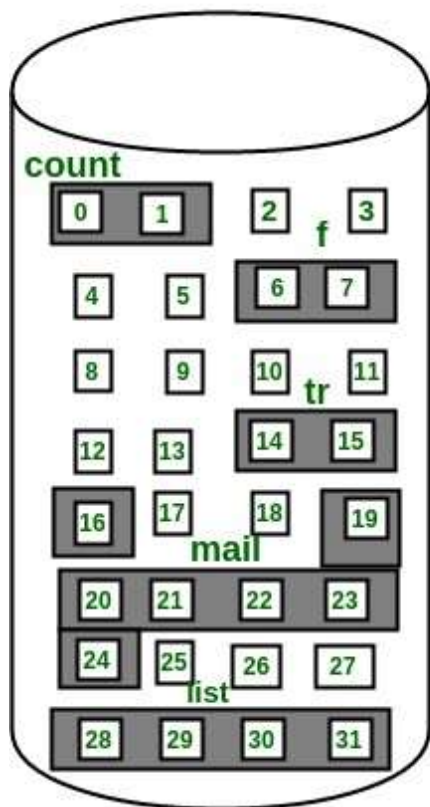
1. Contiguous Allocation

In this scheme, each file occupies a contiguous set of blocks on the disk. For example, if a file requires n blocks and is given a block b as the starting location, then the blocks assigned to the file will be: $b, b+1, b+2, \dots, b+n-1$. This means that given the starting block address and the length of the file (in terms of blocks required), we can determine the blocks occupied by the file.

The directory entry for a file with contiguous allocation contains

- Address of starting block
- Length of the allocated portion.

The file 'mail' in the following figure starts from the block 19 with length = 6 blocks. Therefore, it occupies 19, 20, 21, 22, 23, 24 blocks.



Directory

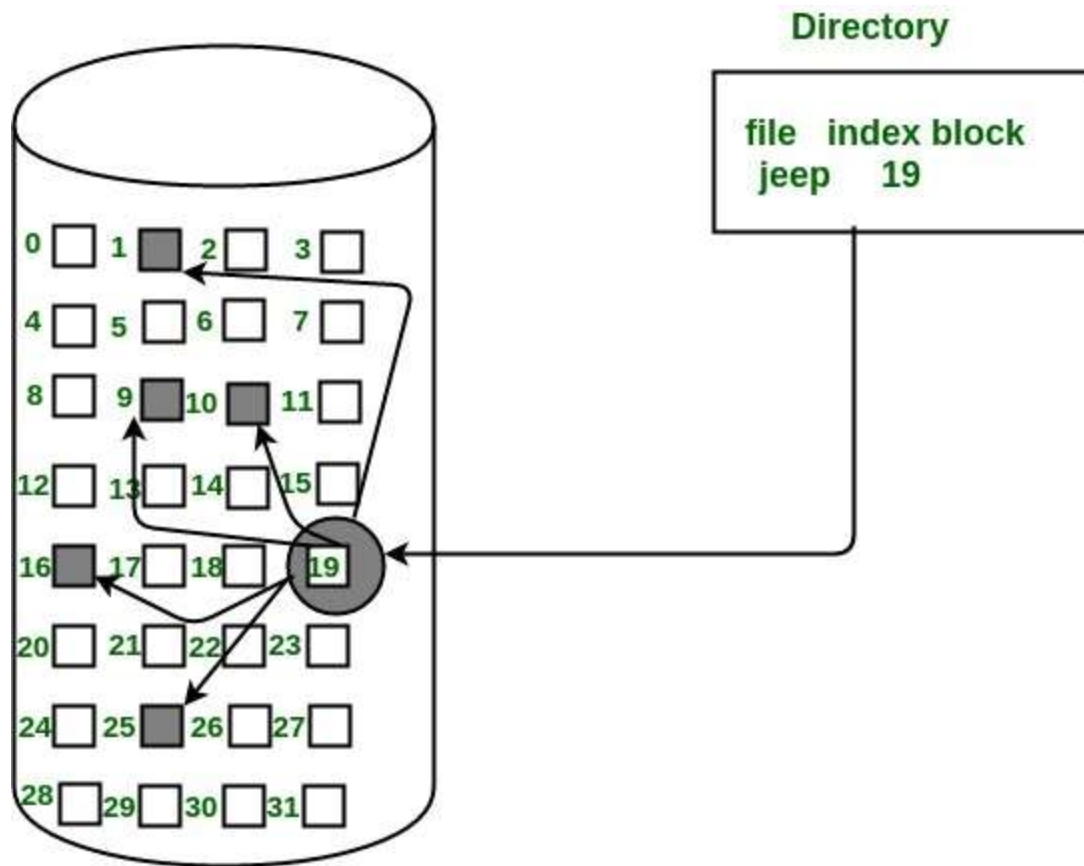
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

2. Linked List Allocation

In this scheme, each file is a linked list of disk blocks which need not be contiguous. The disk blocks can be scattered anywhere on the disk.

The directory entry contains a pointer to the starting and the ending file block. Each block contains a pointer to the next block occupied by the file.

The file 'jeep' in following image shows how the blocks are randomly distributed. The last block (25) contains -1 indicating a null pointer and does not point to any other block



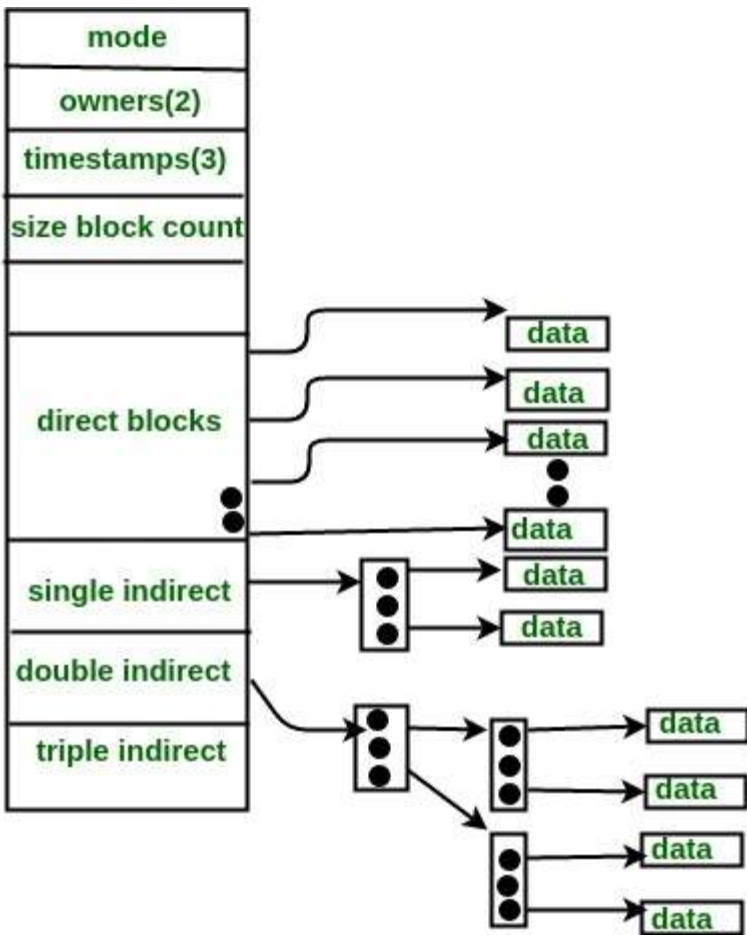
For files that are very large, single index block may not be able to hold all the pointers.

Following mechanisms can be used to resolve this:

Linked scheme: This scheme links two or more index blocks together for holding the pointers. Every index block would then contain a pointer or the address to the next index block.

Multilevel index: In this policy, a first level index block is used to point to the second level index blocks which in turn points to the disk blocks occupied by the file. This can be extended to 3 or more levels depending on the maximum file size.

Combined Scheme: In this scheme, a special block called the Inode (information Node) contains all the information about the file such as the name, size, authority, etc and the remaining space of Inode is used to store the Disk Block addresses which contain the actual file as shown in the image below. The first few of these pointers in Inode point to the direct blocks i.e the pointers contain the addresses of the disk blocks that contain data of the file. The next few pointers point to indirect blocks. Indirect blocks may be single indirect, double indirect or triple indirect. Single Indirect block is the disk block that does not contain the file data but the disk address of the blocks that contain the file data. Similarly, double indirect blocks do not contain the file data but the disk address of the blocks that contain the address of the blocks containing the file data.



Free Space Management in OS

Overview

As we know, there is limited space (hard disk) in our system. So, there should be proper utilization of space or memory available in our system.

The Operating system works to allocate free space to the files when a file is created. It also creates a free void space when a file is deleted from the system. For doing all these tasks and managing spaces in our system, the operating system works with the help of a free space management system and allocates and de-allocates memory spaces simultaneously. In this article, we are going to learn about these concepts.

What is Free Space Management in OS?

There is a system software in an operating system that manipulates and keeps a track of free spaces to allocate and de-allocate memory blocks to files, this system is called a file management system in an operating system". There is a free space list in an operating system that maintains the record of free blocks.

When a file is created, the operating system searches the free space list for the required space allocated to save a file. While deletion a file, the file system frees the given space and adds this to the free space list.



Methods of Free Space Management in OS

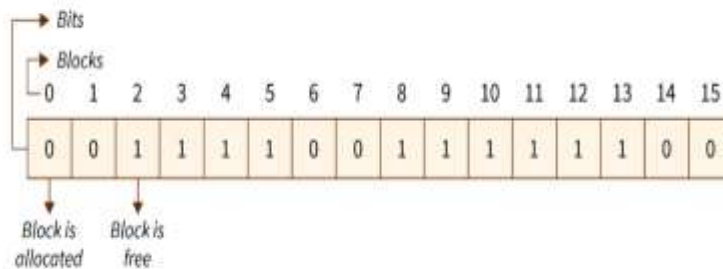
It is not easy work for an operating system to allocate and de-allocate memory blocks (managing free space) simultaneously. The operating system uses various methods for adding free space and freeing up space after deleting a file. There are various methods using which a free space list can be implemented. We are going to explain them below-

Bitmap or Bit Vector

A bit vector is a most frequently used method to implement the free space list. A bit vector is also known as a Bit map. It is a series or collection of bits in which each bit represents a disk block. The values taken by the bits are either 1 or 0. If the block bit is 1, it means the block is empty and if the block bit is 0, it means the block is not free. It is allocated to some files. Since all the blocks are empty initially so, each bit in the bit vector represents 0.

Let us take an example:

Given below is a diagrammatic representation of a disk in which there are 16 blocks. There are some free and some occupied blocks present. The upper part is showing block number. Free blocks are represented by 1 and occupied blocks are represented by 0.



"Free block number" can be defined as that block which does not contain any value, i.e., they are free blocks. The formula to find a free block number is :

[Block number = (number of bits per words)*(number of 0-value word) + Offset of first 1 bit]

We will consider the first 8 bits group (0011110) to constitute a non-zero word since all bits are not 0 here. Non-zero word is that word that contains the bit value 1 (block that is not free). Here, the first non-zero word is the third block of the group. So, the offset will be 3.

Hence, the block number = $8*0+3=3$

Advantages of Bit vector method

- Simple and easy to understand.
- Consumes less memory.
- It is efficient to find free space.

Disadvantages of the Bit vector method

- The operating system goes through all the blocks until it finds a free block. (block whose bit is '0').

- It is not efficient when the disk size is large.

Linked List

A linked list is another approach for free space management in an operating system. In it, all the free blocks inside a disk are linked together in a linked list. These free blocks on the disk are linked together by a pointer. These pointers of the free block contain the address of the next free block and the last pointer of the list points to null which indicates the end of the linked list. This technique is not enough to traverse the list because we have to read each disk block one by one which requires I/O time.



In the above example, we have three blocks of free memory, each represented by a node in the linked list. The first block has 20 bytes of free memory, the second block has 20 bytes of free memory, and the third block has 60 bytes of free memory. The operating system can use this linked list to allocate memory blocks to processes as needed.

Advantage of the Linked list

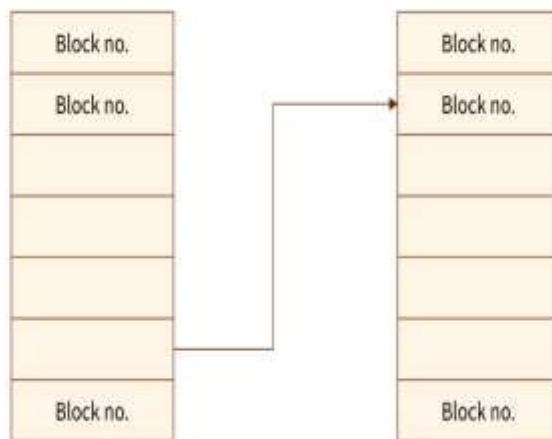
- In this method, available space is used efficiently.
- As there is no size limit on a linked list, a new free space can be added easily.

Disadvantages

- In this method, the overhead of maintaining the pointer appears.
- The Linked list is not efficient when we need to reach every block of memory.

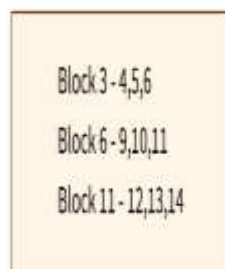
Grouping

The grouping technique is also called the "modification of a linked list technique". In this method, first, the free block of memory contains the addresses of the n-free blocks. And the last free block of these n free blocks contains the addresses of the next n free block of memory and this keeps going on. This technique separates the empty and occupied blocks of space of memory.



Example

Suppose we have a disk with some free blocks and some occupied blocks. The free block numbers are 3,4,5,6,9,10,11,12,13,3,4,5,6,9,10,11,12,13,, and 1414. And occupied block numbers are 1,2,7,8,15,1,2,7,8,15, and 1616 i.e.*i.e.*, they are allocated to some files.



When the "grouping technique" is applied, block 3 will store the addresses of blocks 4, 5, and 6 because block 3 is the first free block. In the same way, block 6 will store the addresses of blocks 9, one 0, and one 1 because block 6 is the first occupied block.

Advantage of the Grouping method

- By using this method, we can easily find addresses of a large number of free blocks easily and quickly.

Disadvantage

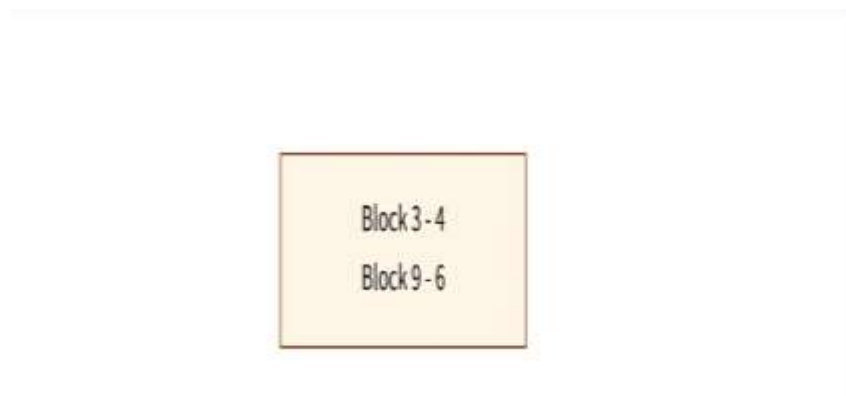
- We need to change the entire list if one block gets occupied.

Counting

In memory space, several files are created and deleted at the same time. For which memory blocks are allocated and de-allocated for the files. Creation of files occupy free blocks and deletion of file frees blocks. When there is an entry in the free space, it consists of two parameters- "address of first free disk block (a pointer)" and "a number 'n'".

Example

Let us take an example where a disk has 16 blocks in which some blocks are empty and some blocks are occupied as given below :



When the "counting technique" is applied, the block number 3 will represent block number 4 because block 3 is the first free block. Then, the block stores the number of free blocks

i.e. - there are 4 free blocks together. In the same way, the first occupied block number 9 will represent block number 10 and keeps the number of rest occupied blocks i.e. - there are 6 occupied blocks as shown in the above figure.

Advantages

- In this method, a bunch of free blocks takes place fastly.

- The list is smaller in size.

Disadvantage

- In the counting method, the first free block stores the rest free blocks, so it requires more space.

Advantages and Disadvantages of Free Space Management Techniques in Operating Systems

Free space management is a critical component of operating systems, aiming to optimize the utilization of storage space. Below are the general advantages and disadvantages associated with these techniques.

Advantages

- **Efficient Use of Storage Space:** These techniques ensure optimal utilization of available space on hard disks and other secondary storage devices, minimizing wastage.
- **Ease of Implementation:** Some methods, like linked allocation, are straightforward and require minimal overhead in terms of processing and memory resources.
- **Faster File Access:** Techniques such as contiguous allocation help in reducing disk fragmentation, leading to quicker access times for files and improved system performance.

Disadvantages

- **Fragmentation:** Certain techniques, particularly linked allocation, can lead to fragmentation of disk space, reducing the efficiency of storage operations.
- **Overhead:** Techniques like indexed allocation may introduce additional overhead, necessitating more memory and processing power to maintain structures like index blocks.
- **Limited Scalability:** Some methods, such as the File Allocation Table (FAT), may not scale well, limiting the number of files that can be efficiently managed on the disk.
- **Risk of Data Loss:** In methods like contiguous allocation, if a file becomes corrupted or damaged, it may be challenging to recover the entire data, leading to potential data loss.

Conclusion

- File management system in an operating system is used to keep track of free spaces to allocate and de-allocate memory blocks to files.
- These space blocks are manipulated when there is a creation or deletion of a file in our system.
- There are various approaches for free space management in os:

1. Bit vector
2. Linked List
3. Grouping
4. Counting