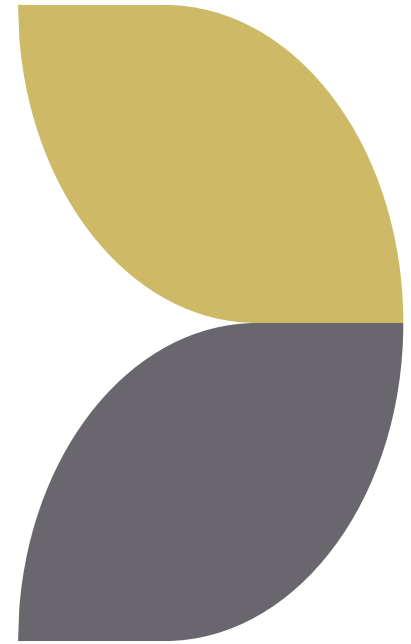


Operating System with Linux as case study

Module 1.2



File System Management



File

- A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks.
- It is a sequence of bits, bytes, lines or records whose meaning is defined by the file's creator and user.
- Files can be stored on disk or other storage and do not disappear when a user logs off.



File Attributes

- **Name** – only information kept in human-readable form
- **Identifier** – unique tag (number) identifies file within file system
- **Type** – needed for systems that support different types
- **Location** – pointer to file location on device
- **Size** – current file size
- **Protection** – controls who can do reading, writing, executing
- **Time, date, and user identification** – data for protection, security, and usage monitoring.

Information about files are kept in the directory structure, which is maintained on the disk

File Management

The main objectives of the file management system are:

- It provides I/O support for a variety of storage device types.
- Minimizes the chances of lost or destroyed data.
- It provides I/O support for multiple users in a multiuser systems environment.



Properties of a File System

- Files are stored on disk or other storage and do not disappear when a user logs off.
- Files have names and are associated with access permission that permits controlled sharing.
- Files could be arranged or more complex structures to reflect the relationship between them.



File Operations

- File is an **abstract data type**
 - Create
 - Write
 - Read
 - Reposition within file
 - Delete
 - Truncate
- $Open(F_i)$ – search the directory structure on disk for entry F_i , and move the content of entry to memory
- $Close(F_i)$ – move the content of entry F_i in memory to directory structure on disk



File Operations

- **Truncating** a file does not remove the file entry from the file system or free up the disk space entirely.
- The file still exists, but its size is reduced, and the removed data is irretrievably lost.

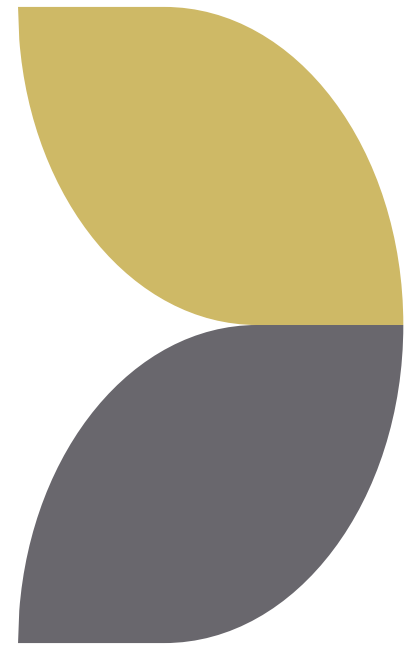
In contrast, **deleting** a file removes the file entry and eventually reclaims the disk space when it is reused by new files.

- It's worth mentioning that the specific behavior of file deletion and truncation may vary depending on the operating system and file system in use, but the general principles described above hold true for most systems.

File Types – Name, Extension

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, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes com- pressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

File Access Methods



File Access Methods

- ❑ Sequential Access
- ❑ Direct Access
- ❑ Indexed Sequential Access

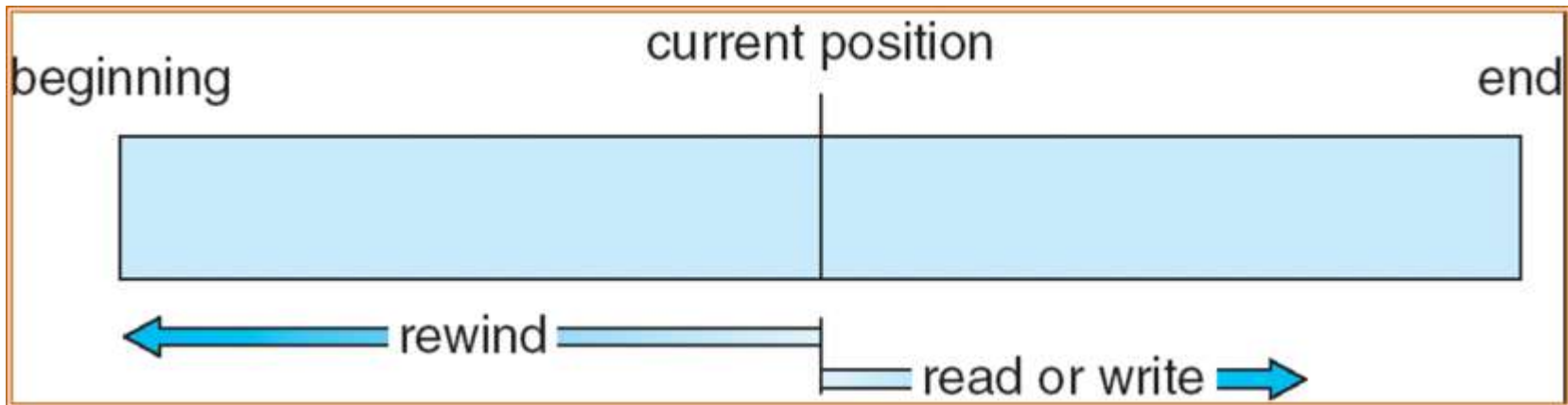
Sequential File Access

- Most of the operating systems access the file sequentially. In other words, we can say that most of the files need to be accessed sequentially by the operating system.
- In sequential access, the OS read the file word by word. A pointer is maintained which initially points to the base address of the file. If the user wants to read first word of the file then the pointer provides that word to the user and increases its value by 1 word. This process continues till the end of the file.



Sequential File Access

- Modern word systems do provide the concept of direct access and indexed access but the most used method is sequential access due to the fact that most of the files such as text files, audio files, video files, etc need to be sequentially accessed.

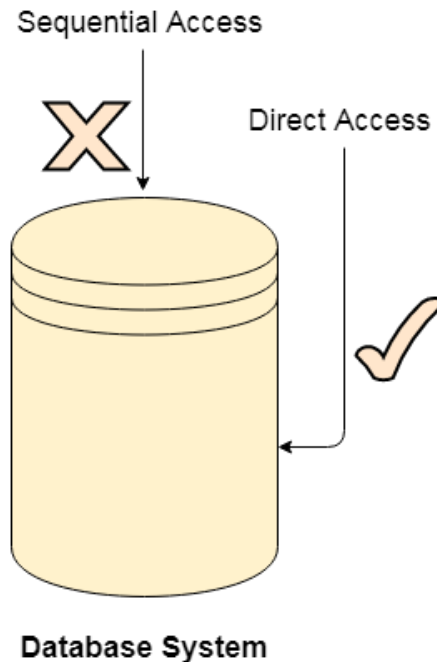


Direct Access

- The Direct Access is mostly required in the case of database systems. In most of the cases, we need filtered information from the database. The sequential access can be very slow and inefficient in such cases.
- Suppose every block of the storage stores 4 records and we know that the record we needed is stored in 10th block. In that case, the sequential access will not be implemented because it will traverse all the blocks in order to access the needed record.

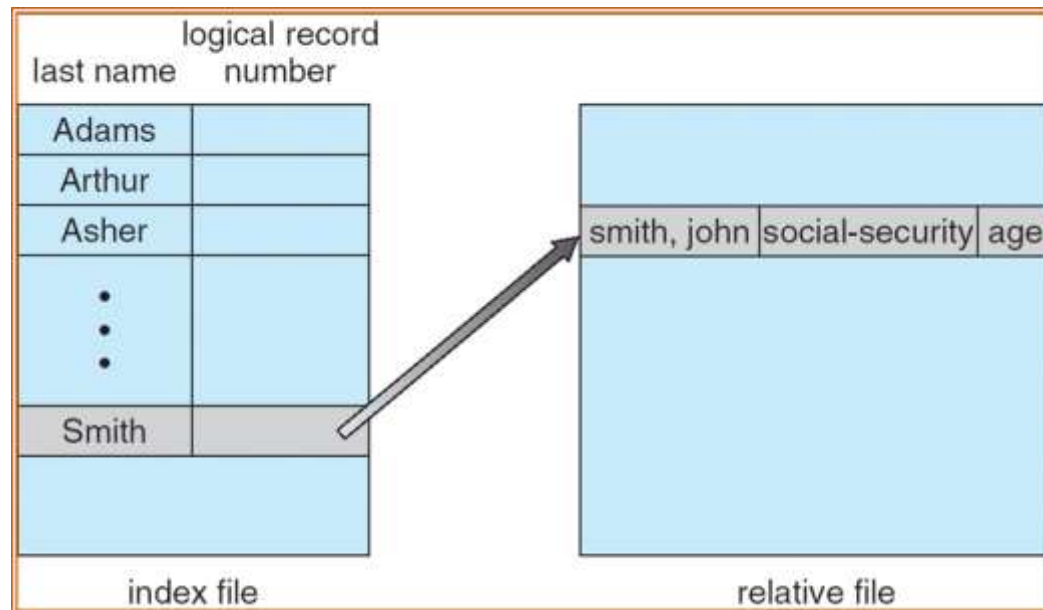
Direct Access

- Direct access will give the required result despite of the fact that the operating system has to perform some complex tasks such as determining the desired block number. However, that is generally implemented in database applications.

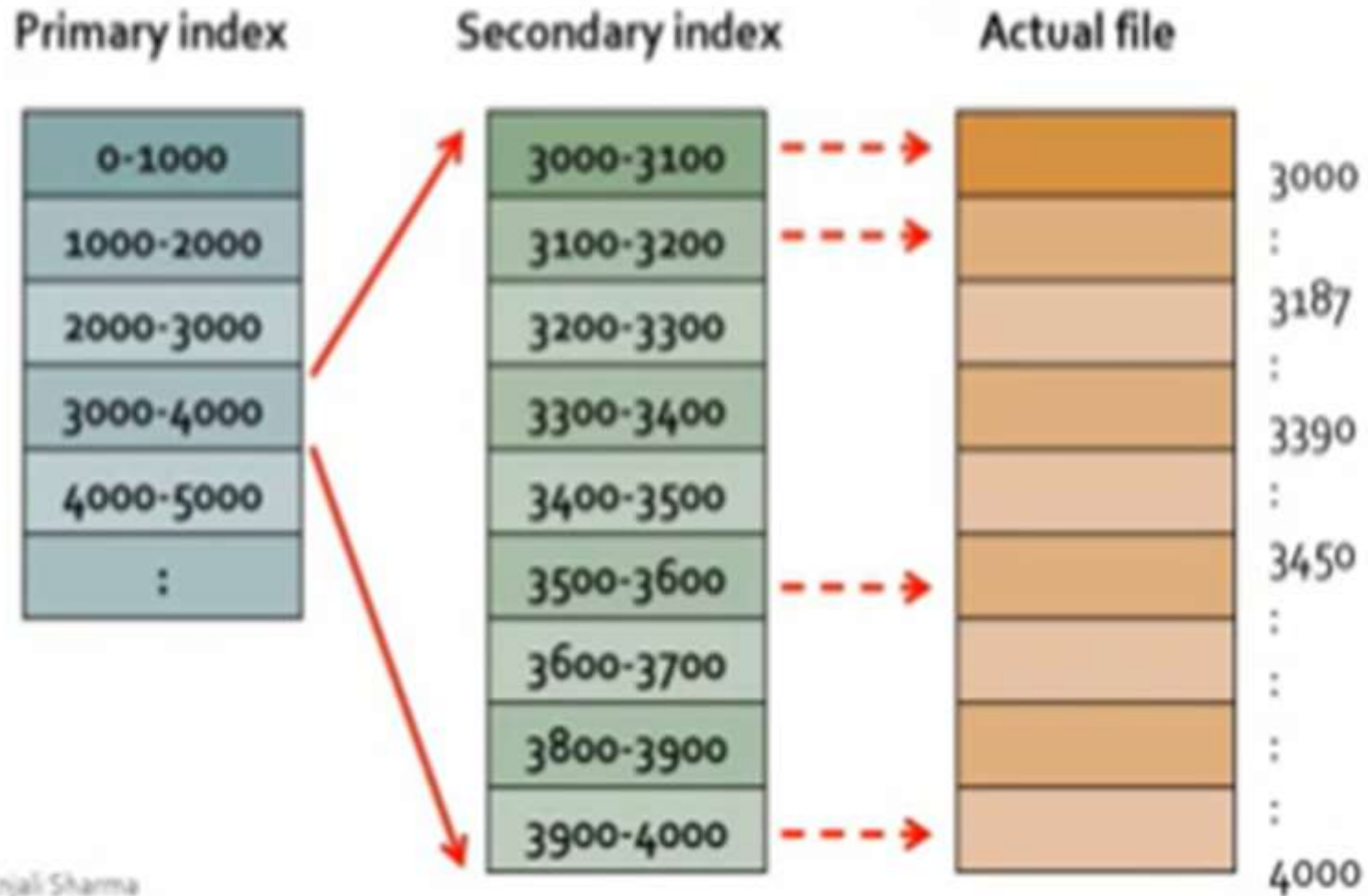


Indexed Sequential Access

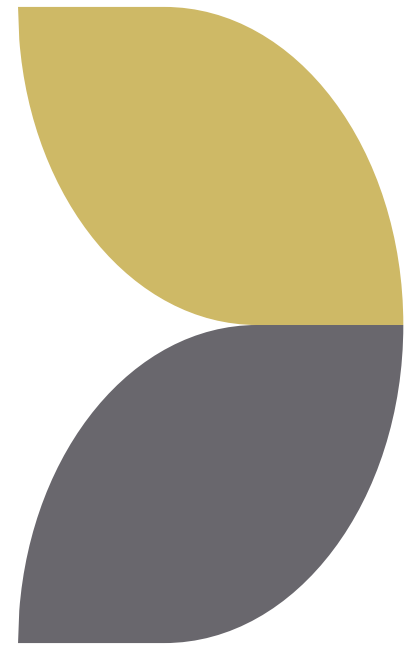
- It is the other method of accessing a file that is built on the top of the sequential access method.
- These methods construct an index for the file.
- The index, like an index in the back of a book, contains the pointer to the various blocks.
- To find a record in the file, we first search the index, and then by the help of pointer we access the file directly.



Indexed Sequential Access



File Allocation Methods



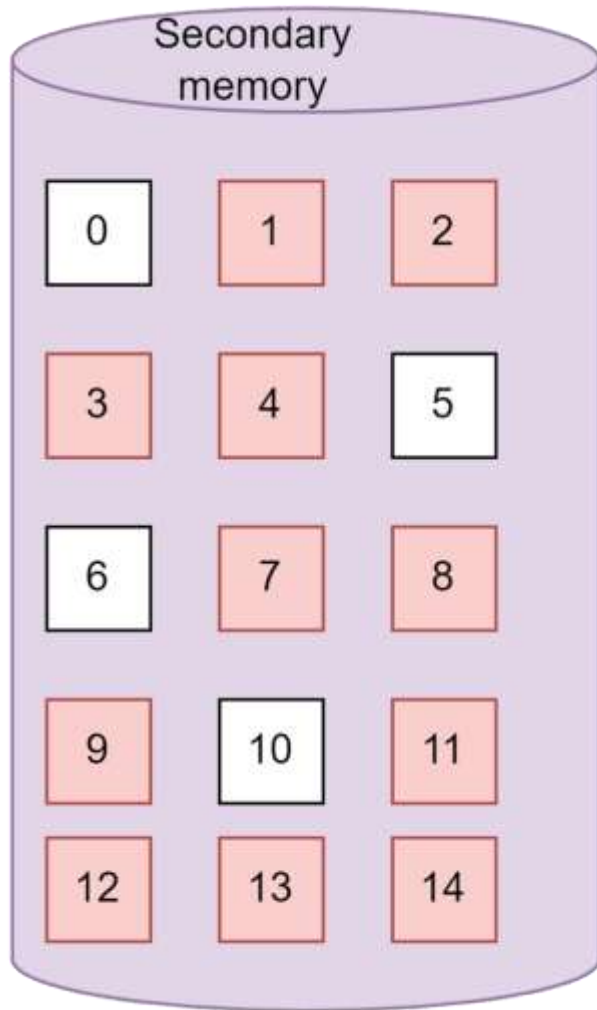
File Allocation Methods

- ❑ Contiguous allocation method
- ❑ Linked allocation method
- ❑ Indexed allocation method
- ❑ File Allocation Table (FAT)
- ❑ Inode

Contiguous allocation method

- In the **contiguous allocation** methods, the files are allocated the disk blocks in a contiguous manner, meaning that if a file's starting disk block address is x , then it will be allocated the blocks with address $x+1$, $x+2$, $x+3$,....., provided the blocks are not already occupied.
- Suppose a file abcd.doc has a starting disk block address of 2, and it requires four such blocks; hence in the contiguous allocation method, the file will be allocated the disk blocks with the address as 2, 3, 4, and 5.

Contiguous allocation method



Directory

File	start	length
os.pdf	1	4
dbms.doc	7	3
dsa.mp4	11	4



Advantages of contiguous allocation

- Since the blocks are allocated in sequential order, therefore it can be accessed in a sequential manner since the starting address and the length is already available in the directory table.
- The block allocation is similar to the array. Given the starting address, we can "jump" to any block address by simply adding the block size to the starting address, just as we do while accessing any block in an array. Hence the contiguous allocation also allows random access to the blocks.
- The seek time is less because of the contiguous allocation of blocks. This makes it very fast.

Disadvantages of contiguous allocation

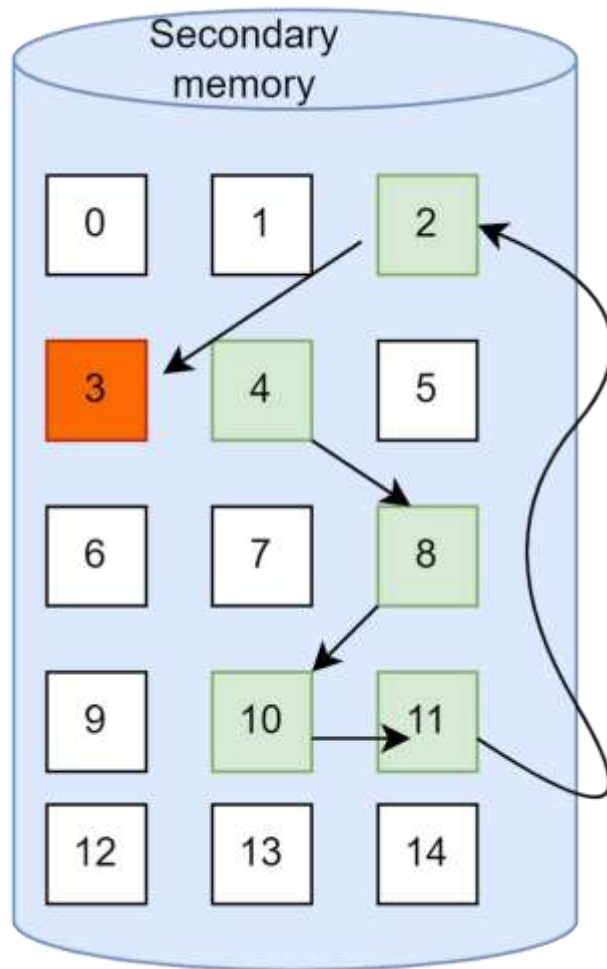
- It suffers from internal fragmentation. Suppose the size of a block is 2KB, but the file that has to be stored is just 1KB. In that case, an extra 1KB remains unutilized and the memory is wasted.
- It suffers from external fragmentation. If there are sufficient blocks available to store a file, but if they are not contiguous, the file cannot be stored.
- The size of the file can be increased only if free contiguous disk blocks are available.

Linked allocation

- The linked allocation works just like the linked list. The problem with contiguous allocation was that memory remained unutilized due to external fragmentation. The solution to the problem was to allocate the disk block in the form of a linked list where every block was a node.
- The blocks are allocated in such a way that every block contains a pointer to the next block that is allocated to the file.



Linked allocation



File	start	end
os.pdf	4	3

Linked allocation

- In the image, the file "os.pdf" has to be allocated some blocks.
- The first block allocated is 4.
- Block 4 will have a pointer to the next block 8, block 8 will have a pointer to block 10, block 10 will have a pointer to block 11, block 11 will have a pointer to block 2, and finally, block 2 will point to 3.
- In this manner, a total of six blocks are allocated to the file in a non-contiguous manner. The ending block (block 3) will not point to any other block.



Advantages of linked allocation

- There is no external fragmentation because blocks can be allocated in random order with the help of pointers. Contiguous allocation is not required.
- File size can be increased, even if the contiguous blocks are not available, provided there are enough blocks to store the file.
- Judicious use of memory.

Disadvantages of linked allocation

- Random access is not allowed since the memory allocation is not contiguous. Therefore with the help of the starting block address, we cannot directly jump to some other block, just as we cannot jump to any node in the linked list with just the head pointer.
- It is relatively slow because of more seek time.
- Every block needs to contain extra information about the pointer to the next block.

Indexed allocation

- Although linked allocation used the memory in an efficient way, it was slower than contiguous allocation.
- There was a need for an allocation method such that the disk blocks are used properly and also the access time is less.
- This is where indexed allocation come as a solution. It can be thought of as a mixture of linked and contiguous allocation that is more efficient in terms of space and time.

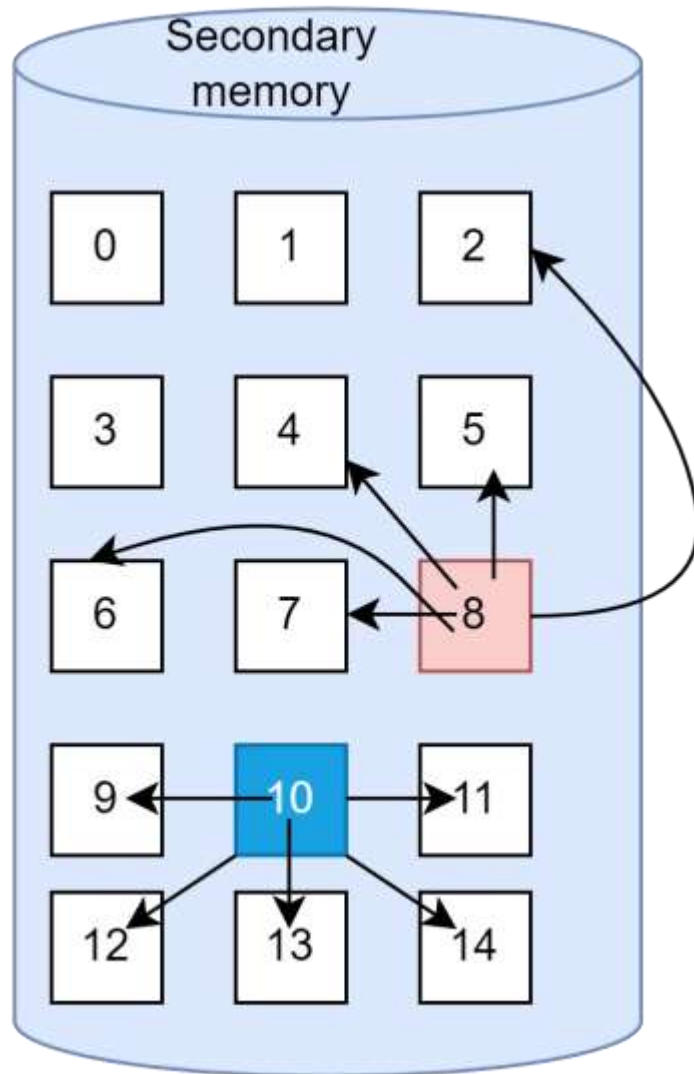


Indexed allocation

- There are index blocks that contain pointers to the blocks occupied by the file.
- Thus every file has its own index block, which is a disk block, but instead of storing the file itself, it contains the pointers to other blocks that store the file.
- This helps in randomly accessing the files and also avoiding external fragmentation.



Indexed allocation



File name	Index block
os.pdf	8
dbms.doc	10






Advantages of Indexed allocation

- No external fragmentation.
- Allows random access to disk blocks.
- Allows direct access, reducing complexity.



Disadvantages of Indexed allocation

- It is very complex.
 - Extra memory for index blocks.
 - Large pointer overhead.
- 

File Allocation Table(FAT)

- The File Allocation Table file system was developed by Microsoft to support efficient storage and user-friendly folder structures.
- Because it employs a table to keep track of the clusters on a storage volume and how the accompanying files and directories connect those clusters, the file system is called File Allocation Table.
- Due to the table's critical role, the name FAT is typically used to refer to the table rather than the file system; however, it's not uncommon to see it used in both contexts within a single resource.

Advantages of File Allocation Table

- In FAT, data is stored using the entire disk block.
- Not all succeeding blocks are lost in case of a faulty disk block.
- There is random access; however, it is not very quick.

Disadvantages of File Allocation Table

- A File Allocation Table entry is required for each disk block.
- The size of the FAT file depends on how many FAT entries there are.
- Increasing the block size will enhance Internal Fragmentation while decreasing the number of FAT entries.

Inode

- In a Unix-style file system, An item such as a file or directory is described by a data structure called an inode (index node).
- Basically, Any Linux file's information, excluding its name and data, is stored in an inode.
- The operating system first looks for the precise and unique inode (inode number) in an inode table whenever a user or program requests access to a file.
- In reality, the inode number obtained from the inode table helps the application or user access the file they need.

Advantages of Inode

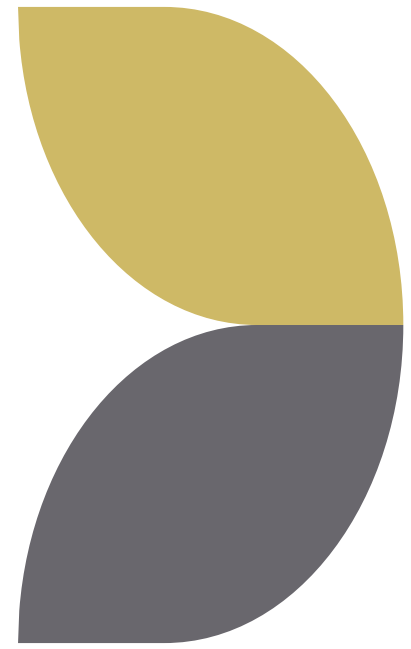
- Filenames are not relevant to inodes.
- An inode's information needs permission to access the file.
- Inode container information regarding the user and group IDs associated with the file.

Disadvantages of Inode

- New files and folders will be rejected once a file system runs out of inodes.
- There is data loss once the inodes are full.



Directory Systems



Directory Structure

- In an operating system, a directory structure refers to the organization and arrangement of directories (also known as folders) and files within a file system.
- It provides a hierarchical framework for storing, accessing, and managing data on a storage device, such as a hard drive or solid-state drive.
- A directory structure typically starts with a root directory, which serves as the highest level of the hierarchy.
- From the root directory, additional directories and subdirectories are created, forming a tree-like structure. Each directory can contain files and additional subdirectories, allowing for the organization of data in a logical manner.

- Whenever a user or a process request for a file, the file system search for the file's entry in the directory and when the match is found, it obtains the file's location from there.

<i>File Name</i>	<i>Type/ size</i>	<i>Location info</i>	<i>Protection Info</i>	<i>Flags</i>	<i>Misc Info</i>



- ***File name*** : The name of the concerned file in the directory, ***Type*** : The kind or category of the file
- ***Location Info*** : Indicates the location where the file is stored.
- ***Protection Info*** : Contains the information whether the file can be accessed by the other user in the system or not.
- ***Flag*** : field contains the kind of directory entry like value ***D*** in Flag field indicates that the file is a directory, value ***L*** indicates that the file is a link, value ***M*** indicates that the file is a mounted file system.
- The ***Misc info*** : Miscellaneous information about the owner of the file, the time of its creation, the time at which the file was modified last.

Types of Directory Structure

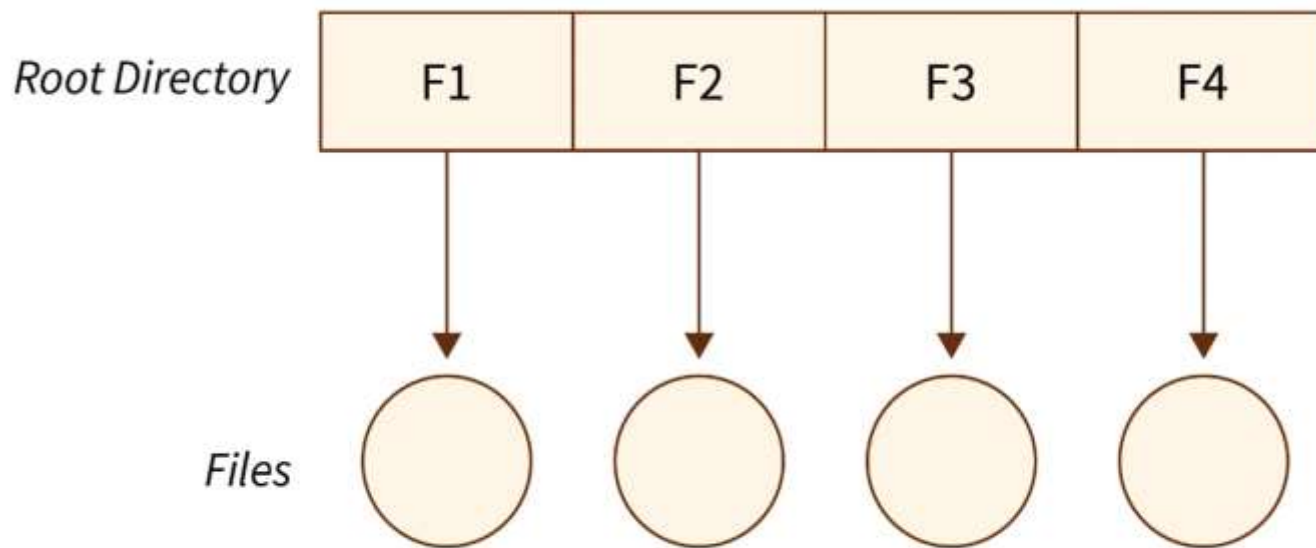
- Single level directory
- Two-level directory
- Tree structure or hierarchical directory
- Acyclic graph directory
- General graph directory structure



Single Level Directory

- The single-level directory structure is the simplest and easiest directory structure out of all the other directories.
- In this directory structure, all the folders/files are contained under the same directory which is called the root directory.
- As the single-level directory structure in gathering all the files under one directory or the root directory, this makes it easy to support and understand.





The Advantages:

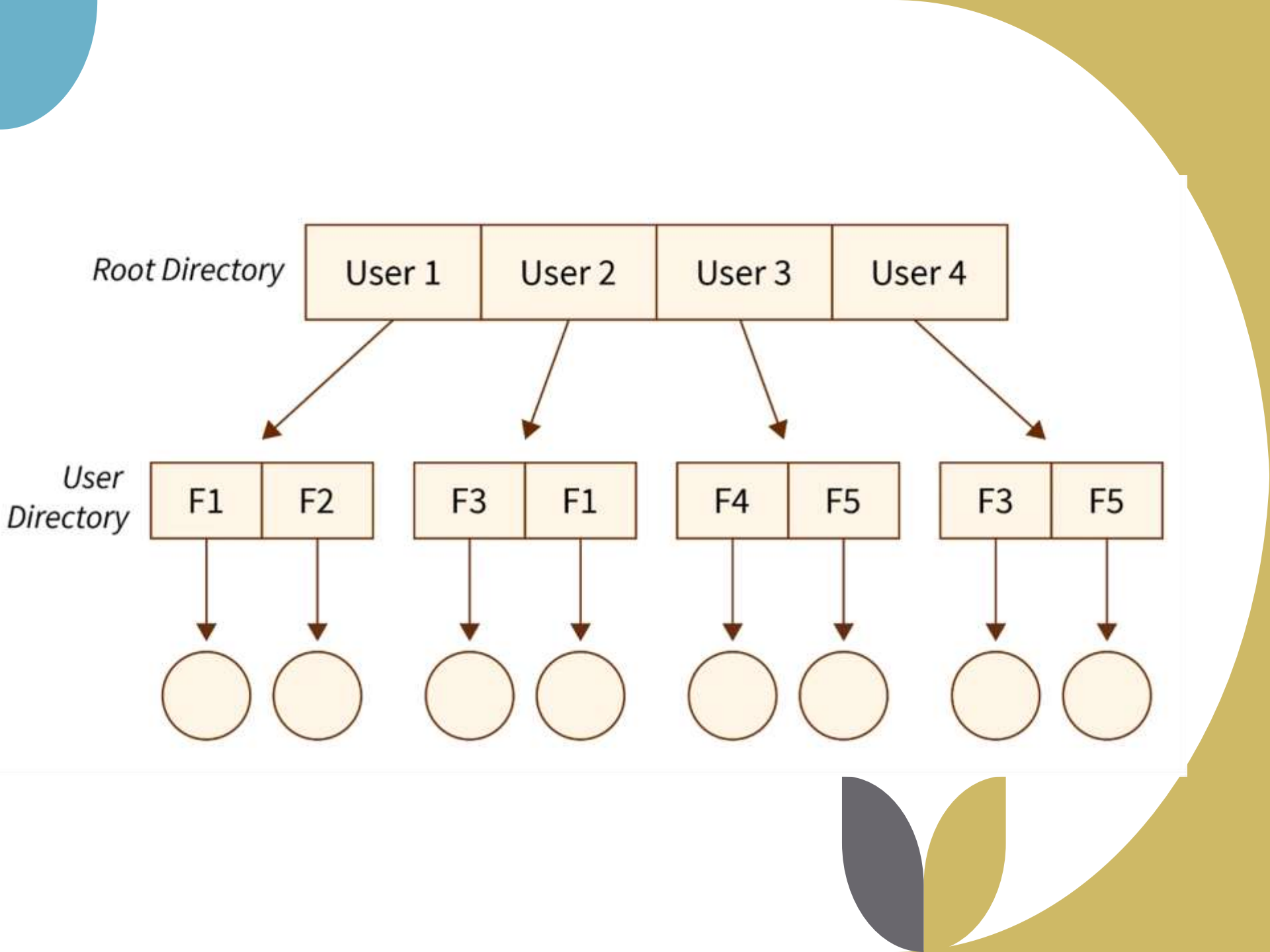
- The implementation of a single-level directory structure is simple and easier as compared to other directory structures in OS.
- If the file size is smaller, then the searching of such files with the single-level directory structure becomes simpler.
- The single-level directory structure allows the operations such as searching, creation, deletion, and updating as well.

The Disadvantages:

- As several users can log in at the same system for logging their files maintaining a unique name becomes difficult leading to a collision. This also means that if the file with the same name is created then the old file will get destroyed first, then the new file (having the same name) created will be replacing it.
- If the size of the files is bigger then searching the files in one root directory of the single-level directory structure will become time taking and hence difficult.
- The single-level directory structure restricts the grouping of the same type of files together.

Two-Level Directory

- The two-level directory structure in OS offers a unique solution to the problem caused by single-level that is, this directory structure it gives each user the right to have their own user files directory commonly called User File Directory(UFD).
- The User File Directory or UFDs has a similar structure as that of the single level, but each UFD lists only the files of a single user who owns that UFD.
- To root all the UFDs, the system's Master File Directory or (MFD) searches whenever a new user id's logged into the directory structure.



The Advantages:

- In the two-level directory structure in OS different users have the right to have the same directory as well as a file name as the user has its own USD which can give a filename that can match other users but won't cause an issue.
- We can also see that searching for files become much simpler.
- As we have a user-defined directory this also provides privacy related to files stored as no user can enter the other user's directory without permission.
- In a two-level directory structure we cannot group the files which are having the same name into a single directory for a specific user.

The Disadvantages:

- In a two-level directory structure a user is not allowed to share files with other users.
- We also find that scalability is not present in a two-level directory structure as two files of the same type cannot be grouped together in the same user.
- Here users cannot create subdirectories only one user file directory can be defined under one master file directory.



Tree Level Directory

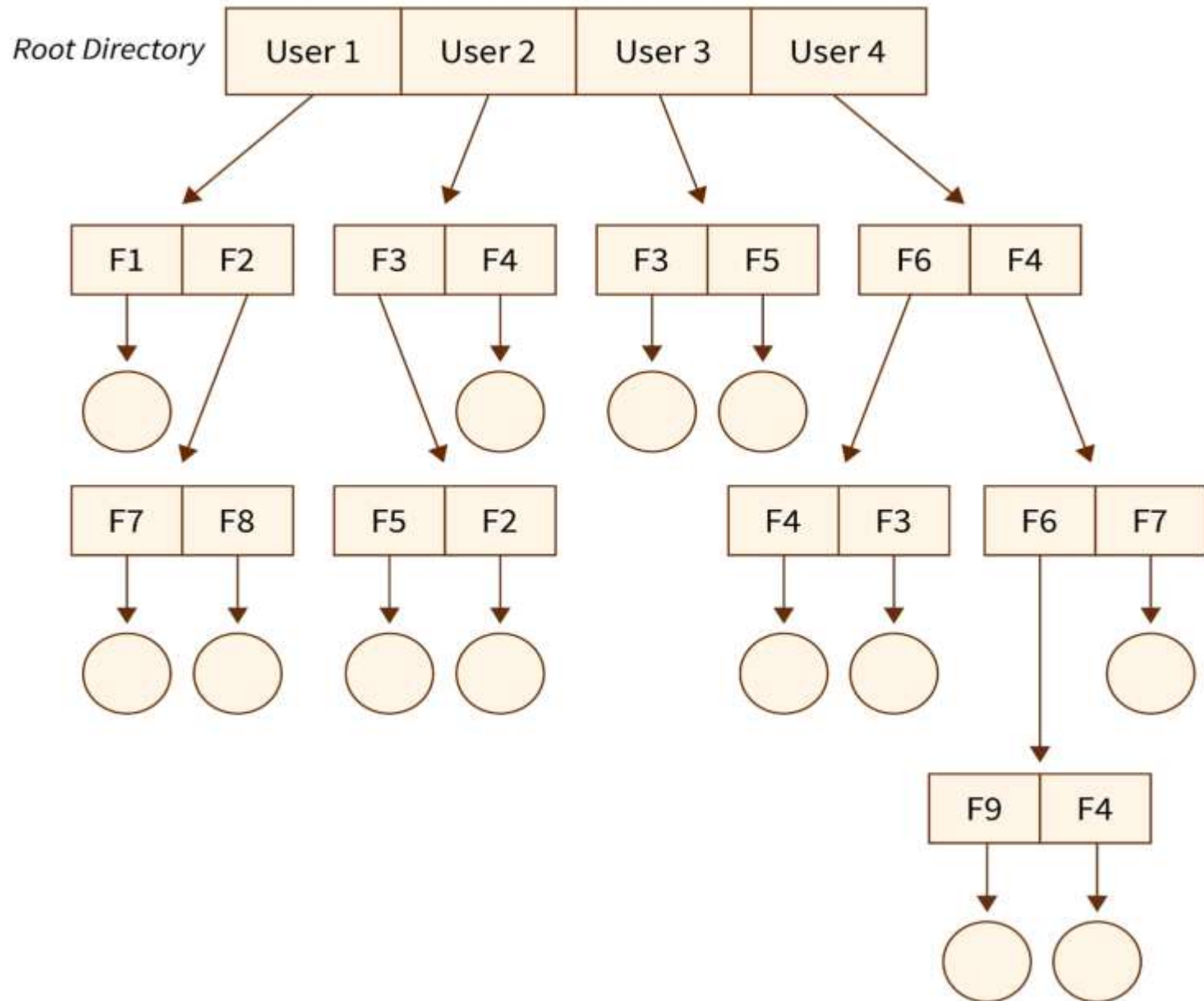
- As observed in the two-level directory structure in OS the drawback of users not having the ability to create sub-directories is resolved with The Tree-structured directory structure in OS coming into the picture.
- The Tree-Structured directory structure in OS is said to be the most common directory structure among users as it gives the users the **capability to create sub-directories under their defined directory.**



- The tree-structured directory structure has separate parent directories for the sub-directories owned by each of their specific users and the parent directories of the users are all under the master-root directory which makes it a tree structure.
- This helps in total separation between the users which provides complete naming freedom and privacy to users' information.
- The system administrator/ UFD admin only has full access to the root directory.



Tree Level Directory



The Advantages:

- In the Tree-structured directory structure searching is quite effective where we use the current working concept that is we can access the file by using two kinds of paths that are either absolute or relative.
- Here we can group the same type of files into one directory.
- In this directory structure the chances of collision of names/types etc are less and hence we can say that the directory structure in OS is scalable.



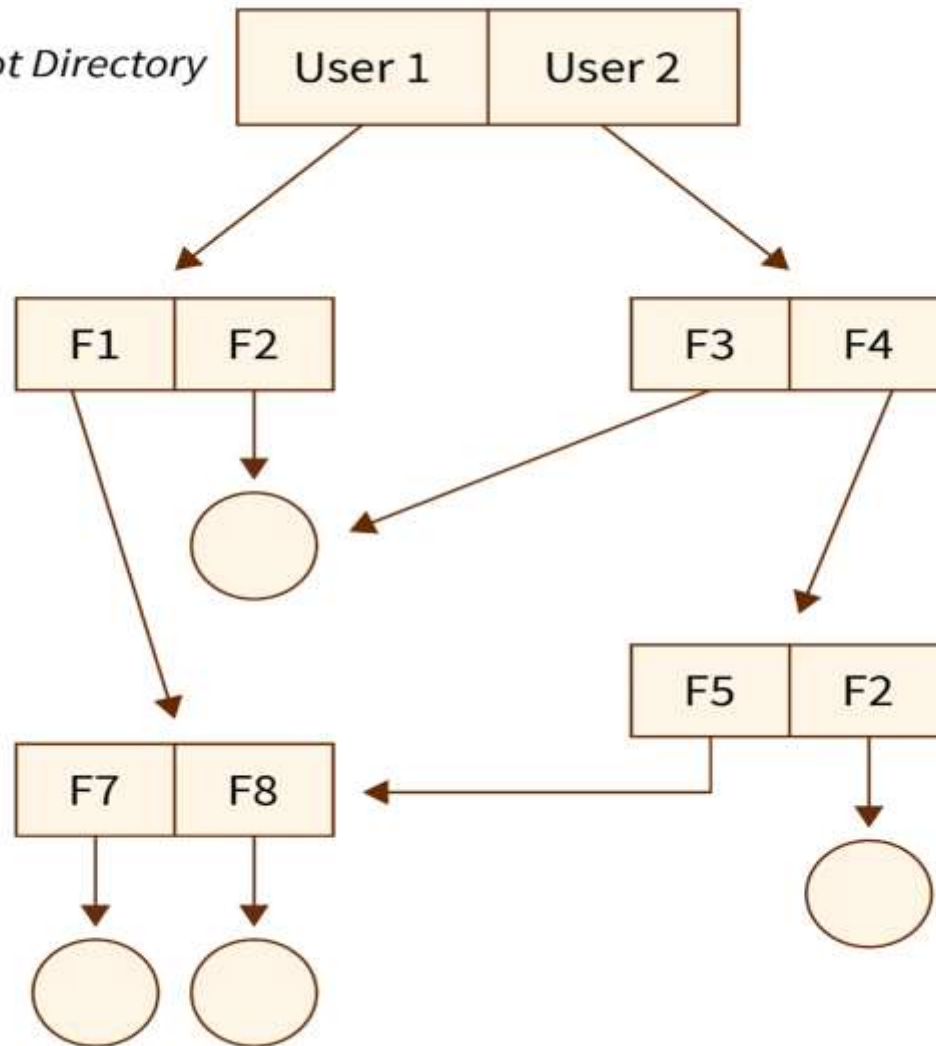
The Disadvantages:

- In the tree-structure directory structure in OS the files cannot be shared between users. Also, the users cannot modify/update the root directory of other users.
- This directory structure in OS as we have to go under multiple directories to access a file we can say that it is said to be inefficient.
- Here each file does not fit into the hierarchal model and so we have to save the files into multiple directories.

Acyclic Graph Directory

- In the Acyclic Graph directory structure can be defined as the directory structure which allows a directory or a file to have multiple parent directories.
- So that it can be a shared file in a directory that gets pointed by the other user directories which if has the access to that shared file via the links provided to it.
- **A file can be shared by another directory, which is not possible in tree structured directories.**
- It is often said to be a natural generalization of the tree-structured directory.

Root Directory



The Advantages:

- In the Acyclic Graph directory structure in OS we can share files between users.
- Here we can search the files easily as compared to the tree-structured directory structure as here we have different-different paths to one file.

The Disadvantages:

- In the Acyclic Graph directory structure in OS as we can share the files via linking, so there are chances that in the case when we want to delete a file in a directory it may create a problem.



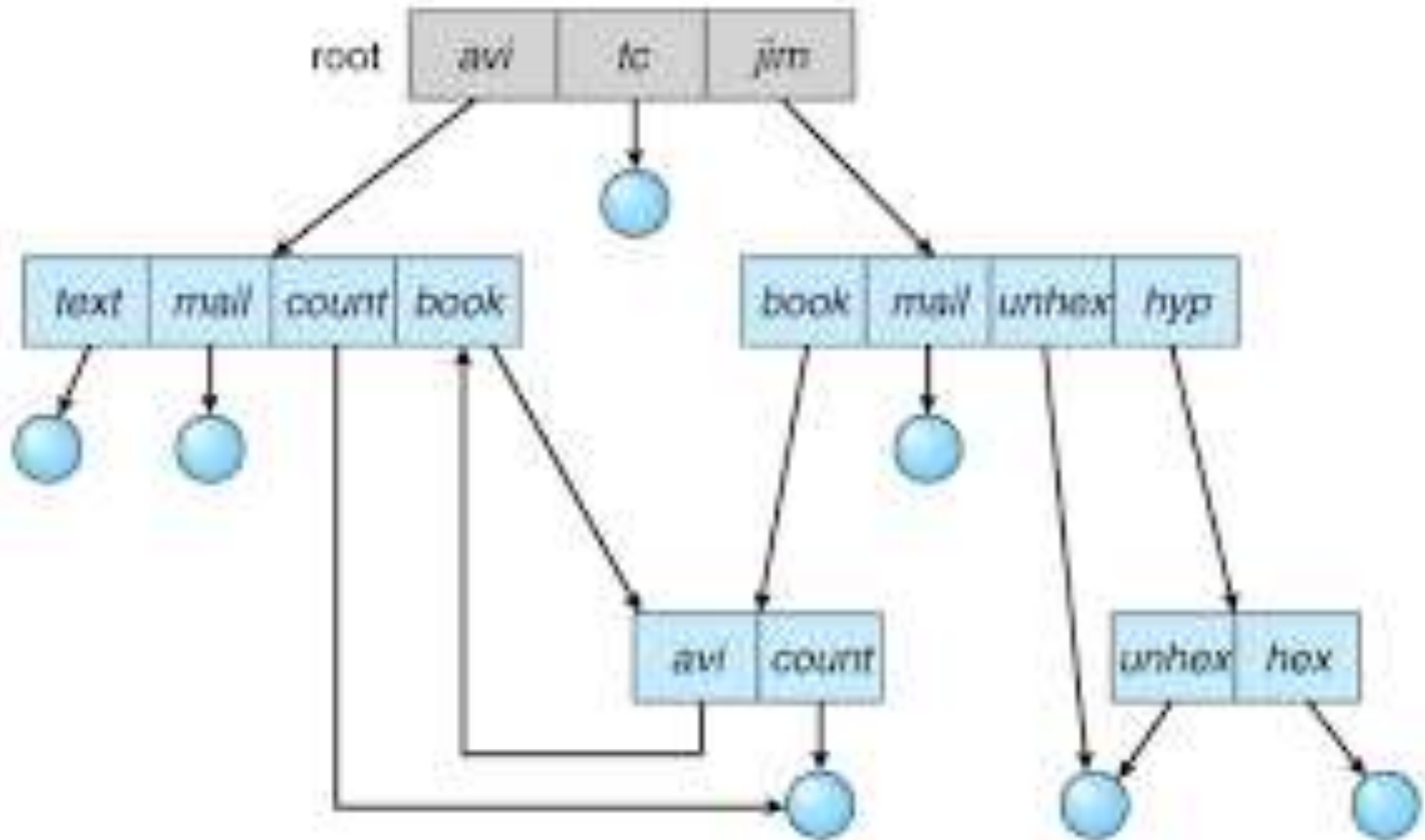
- Also, even if the link is a soft link then after deleting the file we are left with a dangling/suspended point of the link, But in the case of hardlink, when we delete a file we have to vanish all the references associated with it, which can lead to issues associated with referring back to files in case a requirement arises.



General Graph Directory Structure

- In this type of directory, within a directory we can create cycle of the directory where we can derive the various directory with the help of more than one parent directory.
- In this structure the file paths or paths can be categorized into two broad categories to locate the files in the directory structure as below :
 1. **The Absolute Path:** Here, the path of the desired files can be determined by considering the root directory as the base directory.
 2. **The Relative Path:** Here, the path of the desired files can be determined by two choices that are, either the file which needs to be retrieved from its directory is considered the base directory or the user's directory is considered as the base directory.

General graph directory structure



The Advantages:

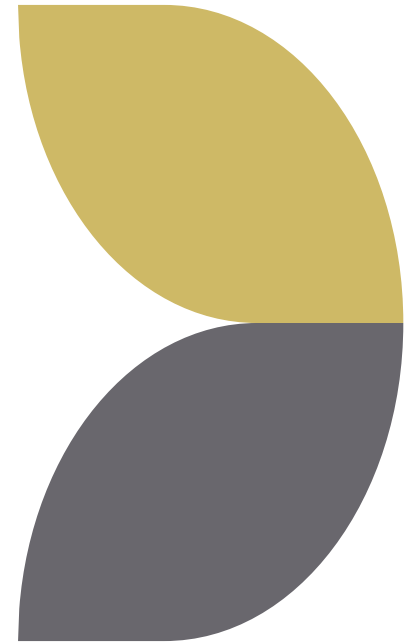
- The General Graph directory structure allows the cycle or creation of a directory within a directory.
- This directory structure is known to be a flexible version compared to other directory structures.

The Disadvantages:

- The main issue which can overpower this structure is to calculate the total size or space that the directories will take up.
- As this directory structure allows the creation of multiple sub-directories a lot of garbage collection can be required.
- If compared to the other directory structure in OS the General Graph directory structure is a costly structure to be chosen.

- The Single-level directory structure in OS is the simplest and easiest directory as all the files are under one directory specifically the root directory.
- The Two-level directory structure in OS has each of its users the right to create a directory directly inside the root directory.
- In the Tree-Structured directory structure in OS among users as it gives the users the capability to create sub-directories under their defined directory which helps them in compartmentalizing their files/folders well.
- In the Acyclic Graph directory structure in OS can be defined as the directory structure which allows a directory or a file to have multiple parent directories
- In the General Graph directory structure in OS users have the capability to create a cycle of the directory within a directory(ie, the directories can be shared by other directory)

File Protection



File Protection

- File protection in an operating system refers to the various mechanisms and techniques used to secure files from unauthorized access, alteration, or deletion.
- It involves controlling access to files, ensuring their security and confidentiality, and preventing data breaches and other security incidents.



- Proper file protection requires ongoing updates and patches to fix vulnerabilities and prevent security breaches.
- It is crucial for data security in the digital age where cyber threats are prevalent.
- By implementing file protection measures, organizations can safeguard their files, maintain data confidentiality, and minimize the risk of data breaches and other security incidents.



File Protection Mechanisms

- File Permissions
- Encryption
- Access Control Lists (ACLs)
- Auditing and Logging
- Physical File Security

File Permissions

- File permissions are a basic form of file protection that controls access to files by setting permissions for users and groups.
- File permissions allow the system administrator to assign specific access rights to users and groups, which can include **read, write, and execute** privileges.
- These access rights can be assigned at the file or directory level, allowing users and groups to access specific files or directories as needed.
- File permissions can be modified by the system administrator at any time to adjust access privileges, which helps to prevent unauthorized access.

File Permissions

- **Owner permissions** – The owner's permissions determine what actions the owner of the file can perform on the file.
- **Group permissions** – The group's permissions determine what actions a user, who is a member of the group that a file belongs to, can perform on the file.
- **Other (world) permissions** – The permissions for others indicate what action all other users can perform on the file.



File Permissions

rwxr-xr--

- The first three characters represent the permissions for the file's owner. Here, the owner has read (r), write (w) and execute (x) permission.
- The second group of three characters consists of the permissions for the group to which the file belongs. Here, the group has read (r) and execute (x) permission, but no write permission.
- The last group of three characters represents the permissions for everyone else. Here is read (r) only permission.

Encryption

- Encryption is the process of converting plaintext into ciphertext to protect files from unauthorized access.
- Encrypted files can only be accessed by authorized users who have the correct encryption key to decrypt them.
- Encryption is widely used to secure sensitive data such as financial information, personal data, and other confidential information.
- In an operating system, encryption can be applied to individual files or entire directories, providing an extra layer of protection against unauthorized access.

Access Control Lists (ACLs)

- Access control lists (ACLs) are lists of permissions attached to files and directories that define which users or groups have access to them and what actions they can perform on them.
- ACLs can be more granular than file permissions, allowing the system administrator to specify exactly which users or groups can access specific files or directories.
- ACLs can also be used to grant or deny specific permissions, such as read, write, or execute privileges, to individual users or groups.

Auditing and Logging

- Auditing and logging are mechanisms used to track and monitor file access, changes, and deletions.
- It involves creating a record of all file access and changes, including who accessed the file, what actions were performed, and when they were performed.
- Auditing and logging can help to detect and prevent unauthorized access and can also provide an audit trail for compliance purposes.



Physical File Security

- Physical file security involves protecting files from physical damage or theft.
- It includes measures such as file storage and access control, backup and recovery, and physical security best practices.
- Physical file security is essential for ensuring the integrity and availability of critical data, as well as compliance with regulatory requirements.

