University May 8, 1945 GUELMA
faculty of Mathematics, Computer Sciences, Physics and Chemistry
Computer Science department
First year Computer Engineer
Semester 1: Introduction to Operating Systems 1

Chapter 3: LINUX/UNIX DIRECTORIES AND FILES

Dr FERKOUS chokri

File Types

- A file type helps us in identifying the type of content that is saved in the file.
- Linux supports seven different types of files. These file types are:
  1. the Regular file
  2. Directory file
  3. Link file
  4. Character special file
  5. Block special file
  6. Socket file
  7. and Named pipe file.

File Types

- The following table provides a brief description of these file types.

File type	Description
Ordinary or regular files	Contain data of various content types such as text, script, image, videos, etc.
Directory files	Contain the name and address of other files.
Block or character special files	Represent device files such as hard drives, monitors, etc.
Link files	Point or mirror other files
Socket files	Provide inter-process communication
Named pipe files	Allow processes to send data to other processes or receive data from other processes.

File Types

1. Regular or ordinary files

- Regular or ordinary files store data of various content types such as text, audio, video, images, scripts, and programs. There are hundreds of content types. In Linux, regular files can be created with or without an extension.
- To view a complete list of content types and file extensions that your Linux system supports, you can see the `/etc/mime.types` file. The MIME (Multipurpose Internet Mail Extensions) provides a standard designation and classification for file content types.
- Linux file system is mainly designed to store, retrieve, and manage regular files. Typically, all Linux distributions provide a dedicated directory to each user for storing regular files. This directory is known as the user's home directory. A user can store regular files in his home directory.

Note: An extension is a group of characters that is used with the file name to give it a special identity or to group it with files of the similar content type. For easy recognition and processing, files of different content types often use well-known file extensions.

File Types

2. Directory files

- To organize files in a hierarchy, file systems use directories. Directories are also files, but instead of storing data, they store the location of other files. To store the location of files placed in the directory, the directory uses directory entries. Each directory entry stores the name and location of a single file

- Linux file system starts with a directory called / or **root** directory.
- All files and directory files are created under this directory. Except the root directory, each directory has a parent directory.

File Types

4. Special files

- Linux places all special files or device files under the **/dev** directory.
- There are two types of special files: **a character special** file and a **block special file**.
  - **A character special file** represents a device that transfers data in bytes such as a monitor or a printer.
  - **A block special file** represents a device that transfers data in blocks such as a hard drive.

File Types

5. Link files

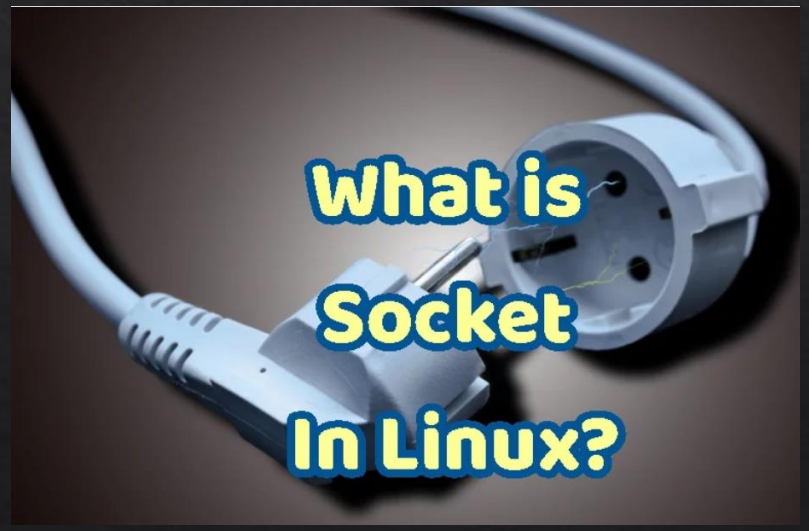
- Link files allow us to use a file with a different filename and from a different location. For this, we use link files. A link file is a pointer to another file. There are two types of links: a **hard link** and a **symbolic** or **soft link**.

- **A hard link** creates a mirror copy of the original file. A hard link cannot be created to a directory or a file on another filesystem.
- **A soft or symbolic link** creates a pointer to the original file. A soft link can be created to a directory or a file on another filesystem.

File Types

- **6. Socket files**

A socket is a communication endpoint that applications use to exchange data. For example, if an application wants to communicate with another application, it connects with the socket of that application.



File Types

- **7. Named pipe files**

- Linux allows us to send the output of any process or command to another process or command as the input. This feature is known as **the pipe**.
- **Pipes** work only when both processes are started by the same user.

File Types

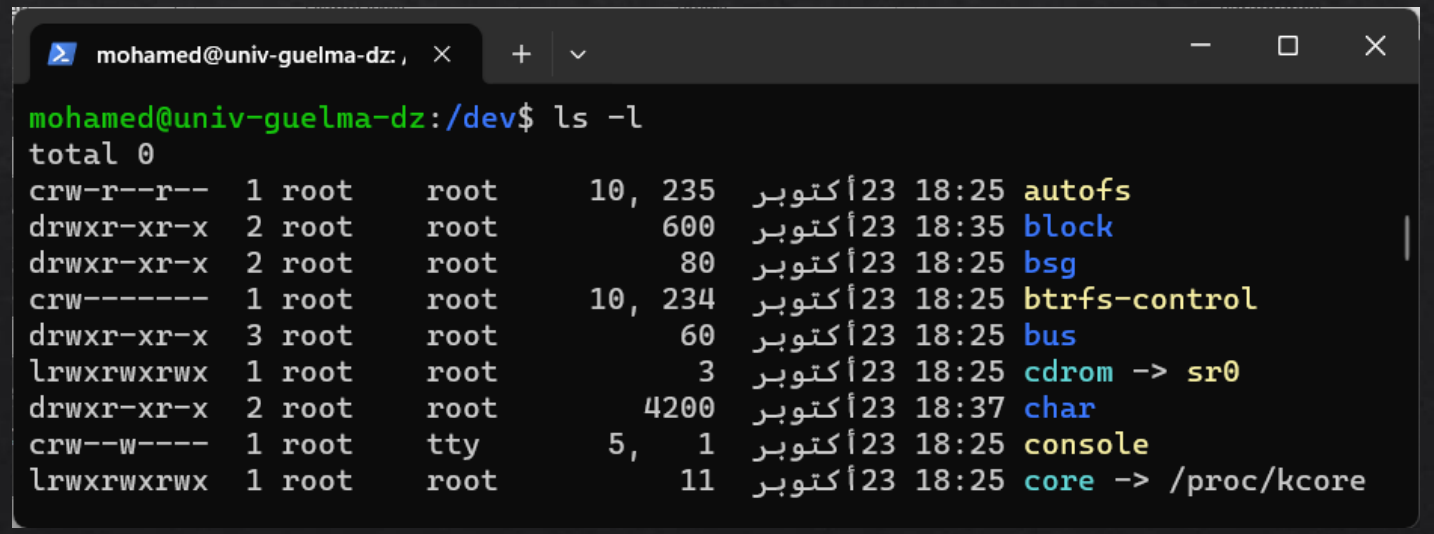
How to identify the type of a file?

- There are many ways to identify the type of a file in Linux. The easiest way is to use the **file** command. To find the type of a file, specify the name of that file as an argument. For example, to know the file type of the file abc, use the following command.

```
$file abc
```

- You can also identify the type of file by looking at the output of the ls -l command. The ls -l command lists the contents of the specified file. For example, the following command lists the content of the current directory.

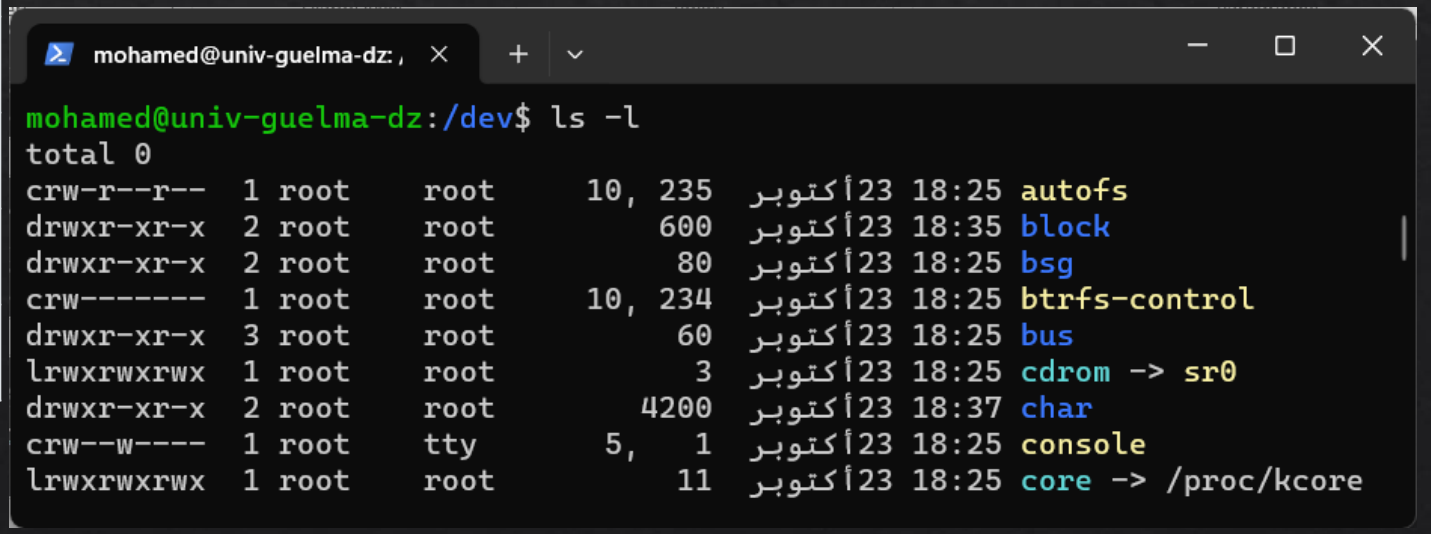
```
$ls -l
```



File Types

- **How to identify the type of a file?**
- In the output listing, the first character of each listing tells the type of the file. The following table lists the symbol of different types of files.

Character	Meaning
-	Regular or ordinary file
d	Directory file
l	Link file
b	Block special file
p	Named pipe file
c	Character special file
s	Socket file



How do hard disk drives work?

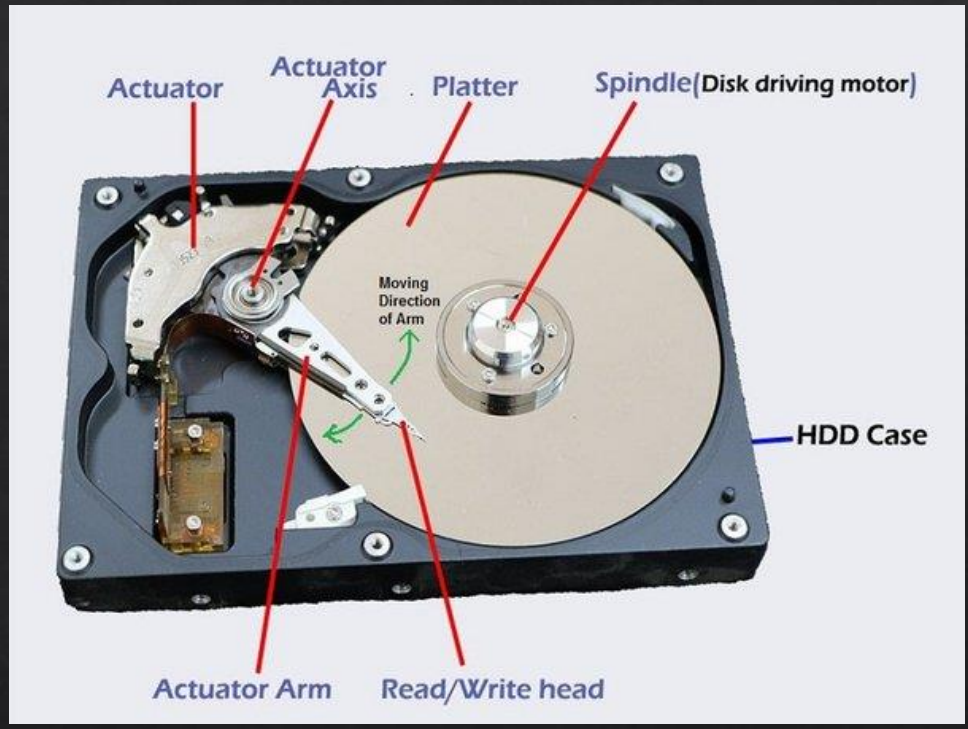
The HDD is made up of several components, including:

1. **Spindle motor**: This motor is responsible for spinning the disks at high speeds.

2. **Read/write head**: This is a small electromagnet that is used to read and write data to the disk. The read/write head is mounted on an arm that can be moved over the surface of the disk to access different parts of it.

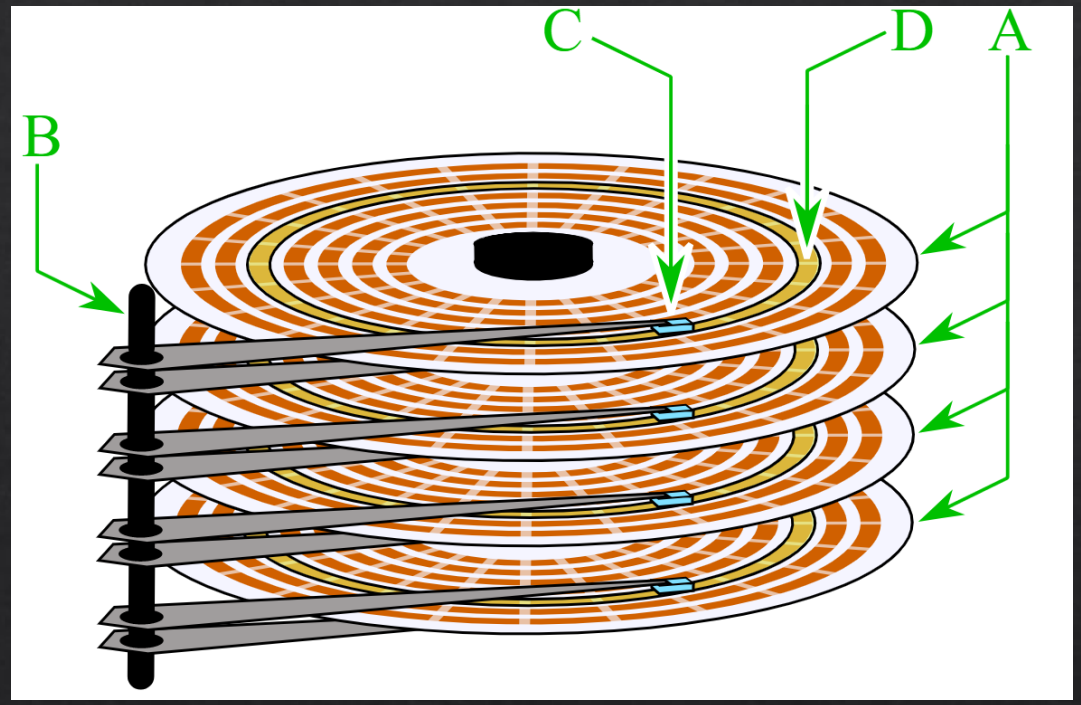
3. **Platter**: The platters are the disks on which data is stored. They are made of a non-magnetic material, such as aluminum, and are coated with a magnetic material.

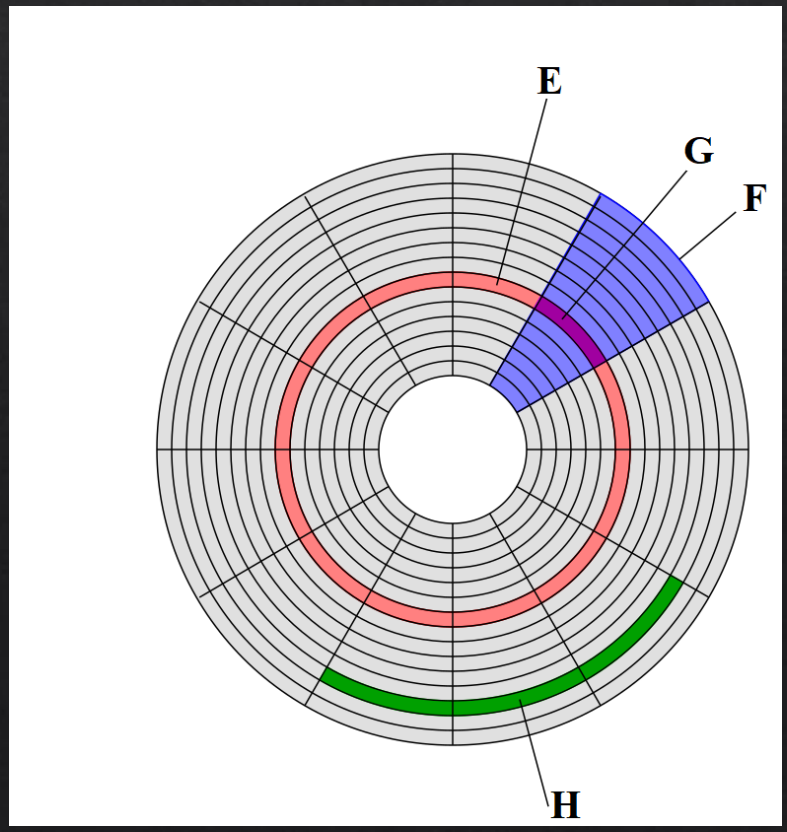
4. **Actuator**: The actuator is an electromechanical device that moves the read/write head over the surface of the disk.



How do hard disk drives work?

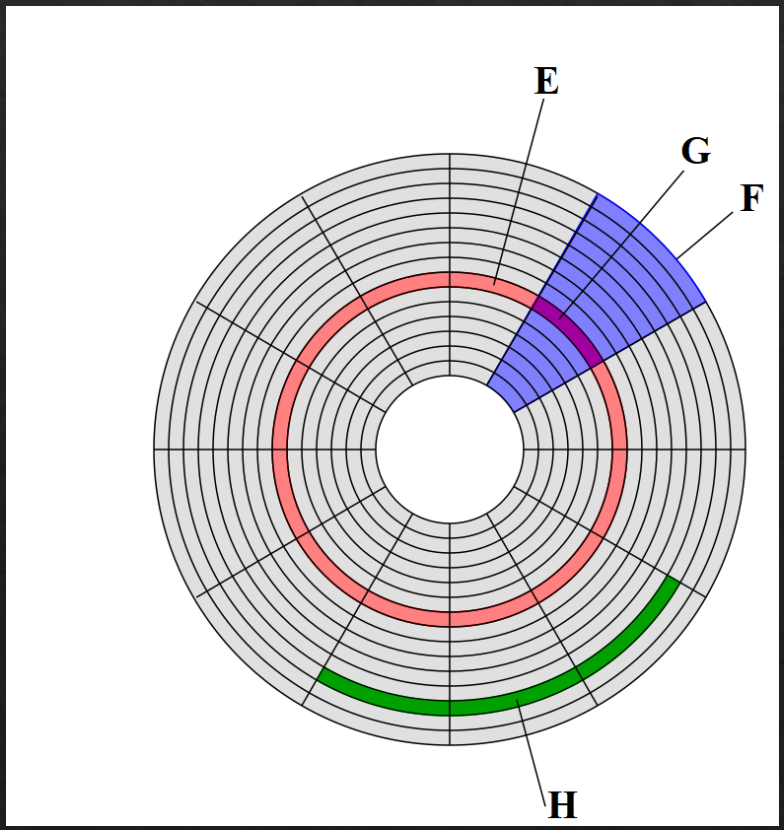
A: Platter;
B: arm;
C: head;
D: cylinder;
E: track;
F: disk sector
G. geometric sector
H. Block (Cluster)





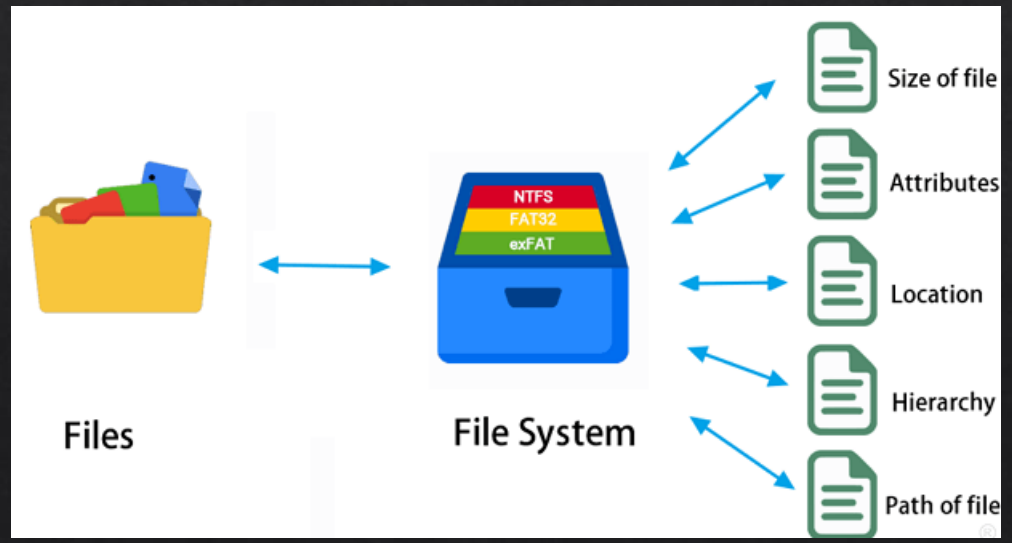
File System

- The block size specifies size that the filesystem will use to read and write data. Larger block sizes will help improve disk I/O performance when using large files, such as databases. This happens because the disk can read or write data for a longer period of time before having to search for the next block.
- On the downside, if you are going to have a lot of smaller files on that filesystem, like the /etc, there the pot ential for a lot of wasted disk space.
- For example, if you set your block size to 4096, or 4K, and you create a file that is 256 bytes in size, it will still consume 4K of space on your harddrive. For one file that may seem trivial, but when your filesystem contains hundreds or thousands of files, this can add up.



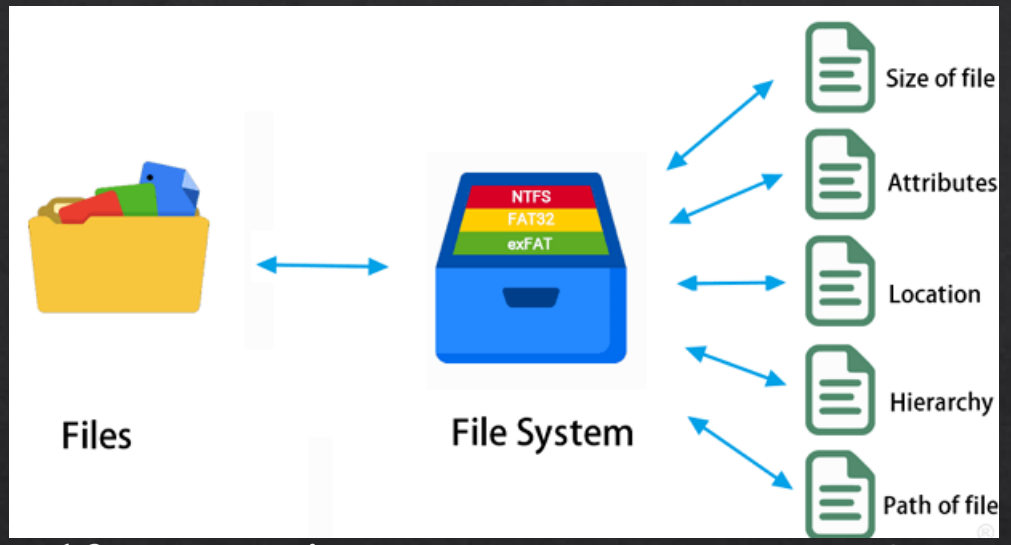
File System

- **File System**, also known as **filesystem** or **fs**, is a method and data structure that the operating system uses to control how data is stored and retrieved.
- By separating the data into pieces and giving each a name, the data is easily isolated and identified.



File System in Linux

- **Windows File System** - FAT, NTFS, exFAT
- **macOS** - HFS, APFS, HFS+
- **Linux** - Ext4, Ext3, Ext2, Btrfs, ZFS, XFS, F2FS, NILFS, and JFS



Ext4 is regarded as the best Linux file system for now.
Ext4 is a direct evolution of the Ext3 file system, which itself was an improvement over Ext2. Ext2 originated in the early ’90s and served as a standard Linux file system for many years.

- High performance.
- Backward compatibility with Ext2/Ext3.
- Support for large file systems.

On ext4, just use tune2fs to check your block size, as follows (change /dev/sda1 to your own device path):

```
$ sudo tune2fs -l /dev/sda7
```

File System

FS	Max File Size / block size	Max Volume Size / block size	Operating System
NTFS	•16EB - 1KB •16TB - 64KB •256TB - 64KB •8PB - 2MB	•256TB - 64KB •8PB - 2MB	•Windows NT3.1 and later •macOS X 10.3 and later (Read-only) •Linux kernel 2.6 and later (read-only) •FreeBSD, NetBSD, OpenBSD(read-only), Chrome OS, Solaris, ReactOS(read-only)
FAT32	•4GB	•2TB - 512 byte •8TB - 2KB •16TB - 4KB	•Win 95OSR2, Win 98, XP, 7, 8, 10, and 11. •macOS •Linux
exFAT	•128 PB	•128 PB	•Windows XP, Vista, 1/8/10/11, Windows Server 2003/2008/2008 R2 •Linux kernal 5.4 and later, FUSE •Mac OS X 6.5 and later
EXT2/3/4	•4TB - 1KB •8TB - 2KB •16TB - 4KB •256PB - 64KB	•4TB - 1KB •8TB - 2KB •16TB - 4KB •256PB - 64K	•Linux kernel 0.96 and later

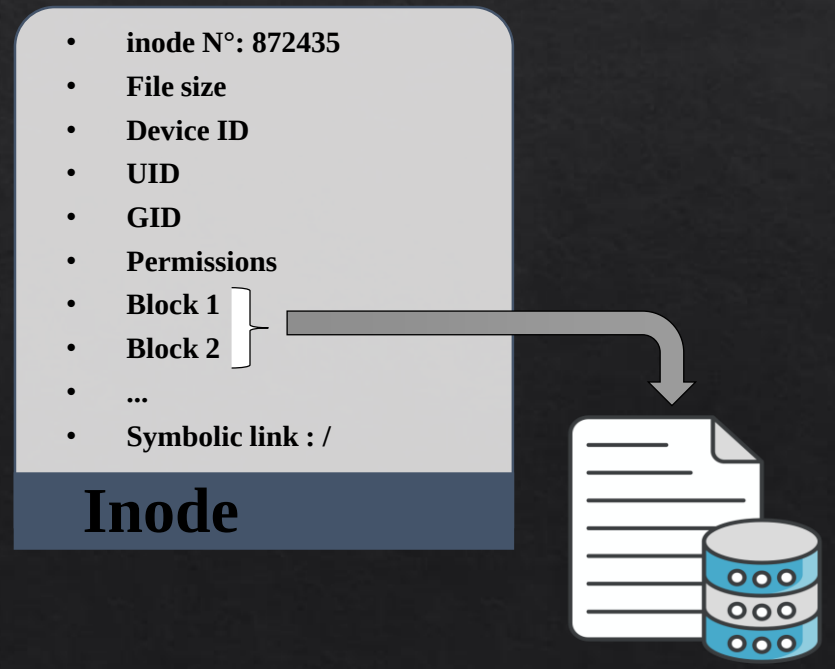
Multiples of bytes

1 KiloBytes (KiB)	=	1024 Bytes
1 MegaByte (MiB)	=	1024 KiBs
1 GegaByte (GiB)	=	1024 MiBs
1 TeraByte (TB)	=	1024 GiBs
1 PetaByte (PB)	=	1024 TBs
1 ExaByte (EB)	=	1024 PBs
1 ZetaByte (ZB)	=	1024 EBs
1 Yottabyte (YB)	=	1024 ZBs

Inode

- An Inode number is a uniquely existing number for all the files in Linux and all Unix type systems.
- When a file is created on a system, a file name and Inode number is assigned to it.
- Generally, to access a file, a user uses the file name but internally file name is first mapped with respective Inode number stored in a table.

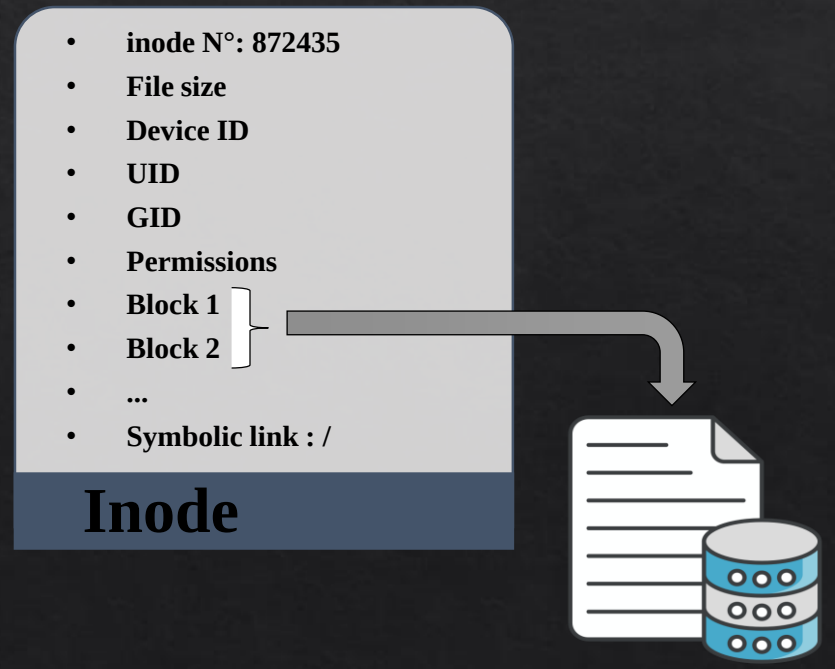
- **Note:** Inode doesn't contain the file name. Reason for this is to maintain hard-links for the files. When all the other information is separated from the file name then only we can have various file names pointing to the same Inode.



Inode table

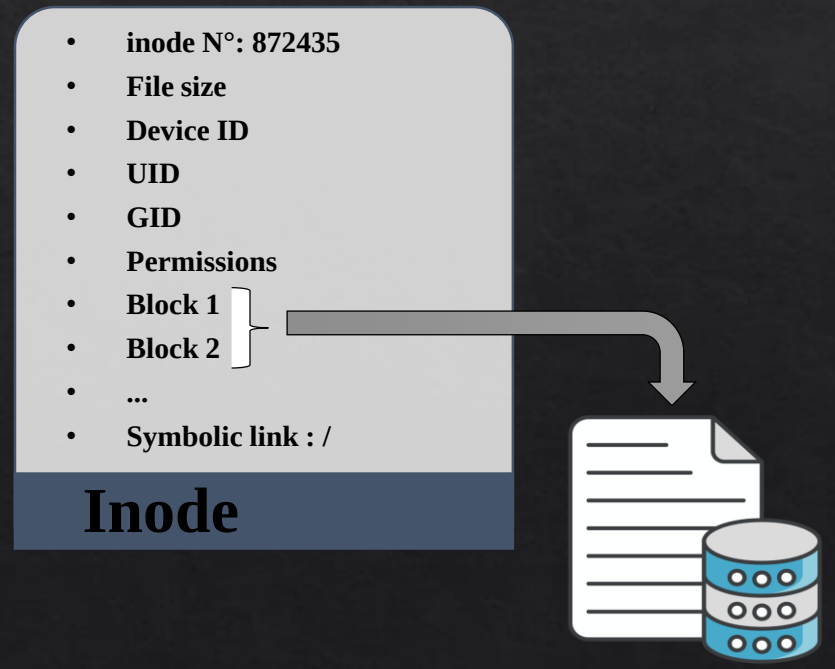
- **Inode Table**

The Inode table contains all the Inodes and is created when file system is created.
The **df -i** command can be used to check how many inodes are free and left unused in the filesystem.



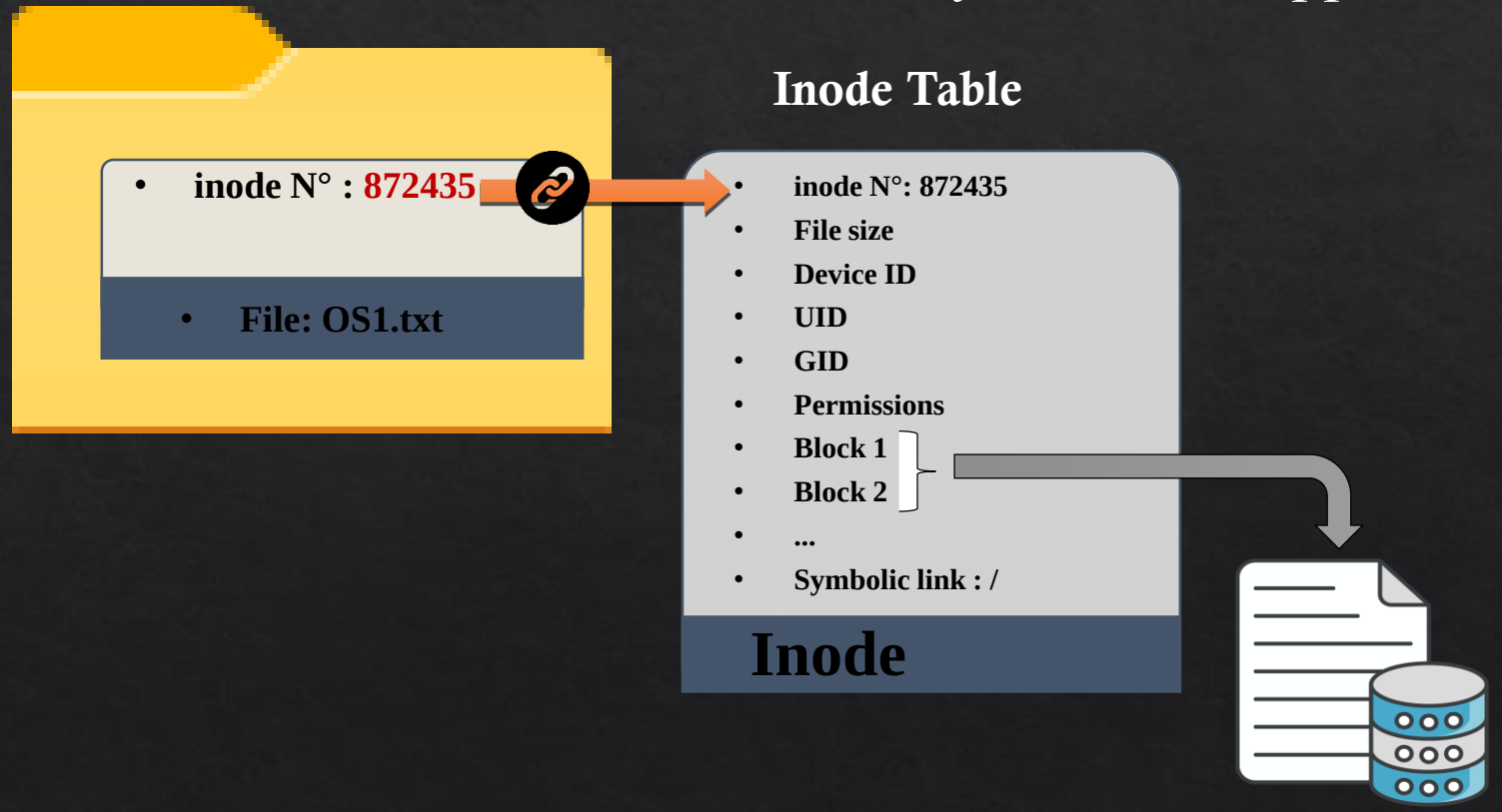
Inode

- Data is stored on your disk in the form of **fixed-size blocks**. If you save a file that exceeds a standard block, your computer will find the next available segment on which to store the rest of your file. Over time, that can get super confusing.
- That's where inodes come in. While they don't contain any of the file's actual data, it stores the file's metadata, including all the storage blocks on which the file's data can be found.



Inode of File in linux

Each inode is identified by an inode number. Therefore, when creating or copying a file, Linux assigns a different inode number to the new file. However, when moving a file, the inode number will only change if the file is moved to a different filesystem. This applies to directories as well.



Inode of File in linux

Inode Contents

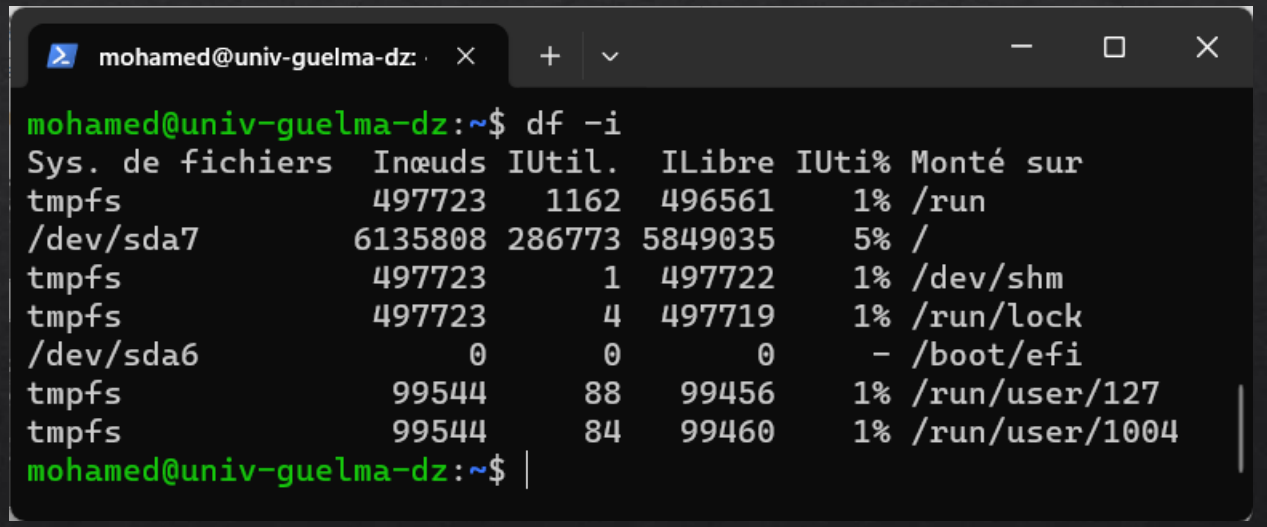
- An Inode is a data structure containing metadata about the files.
- Following contents are stored in the Inode from a file
  1. User ID of file
  2. Group ID of file
  3. Device ID
  4. File size
  5. Date of creation
  6. Permission
  7. Owner of the file
  8. File protection flag
  9. Link counter to determine number of hard links

Example: `$ ls -i`

Inode of File in linux

Inode Table

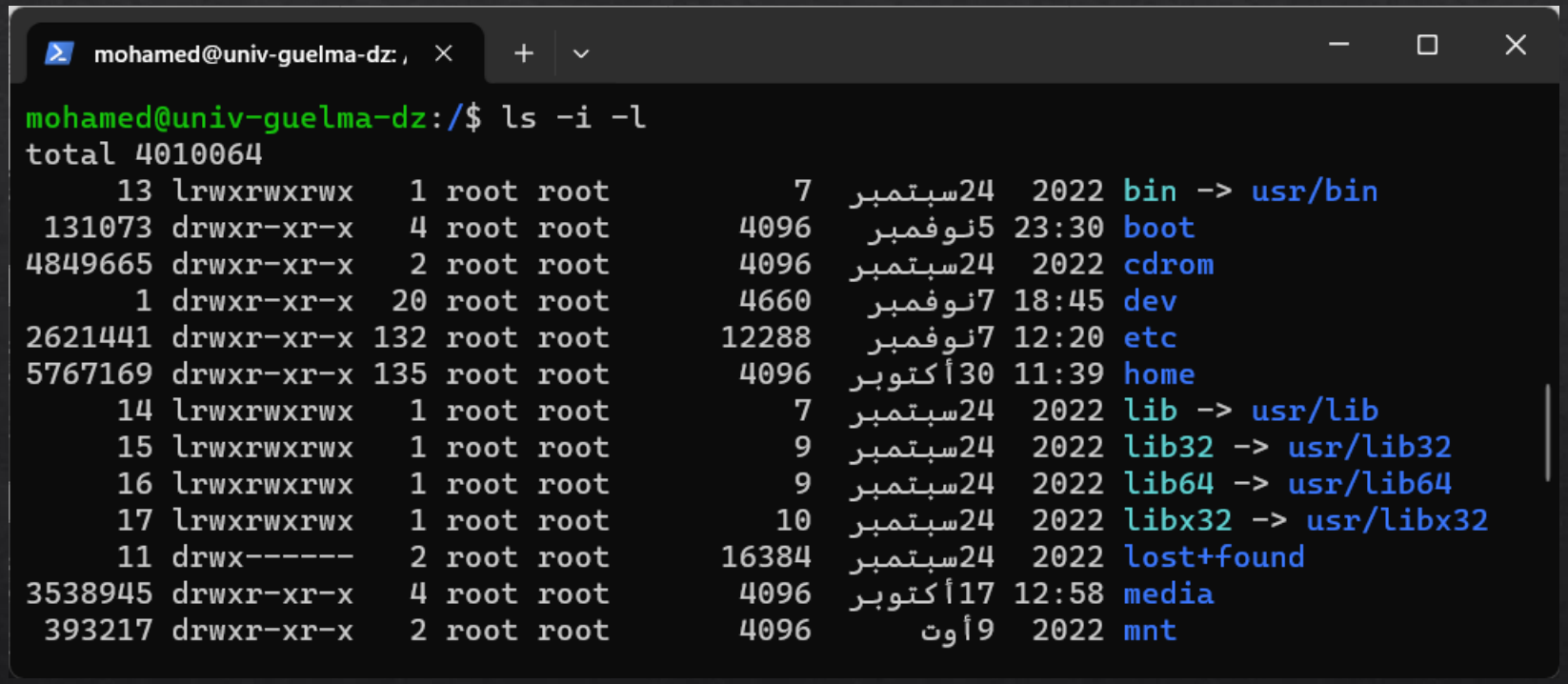
- The Inode table contains all the Inodes and is created when file system is created. The `df -i` command can be used to check how many inodes are free and left unused in the filesystem.



Inode of File in linux

Inode Number

- Each Inode has a unique number and Inode number can be seen with the help of `ls -li` command.



- Look at the above snapshot, Directory Disk1 has the three files and each file has a different Inode number.
- Note: The Inode doesn't contain file content, instead it has a pointer to that data.

Linux File Links

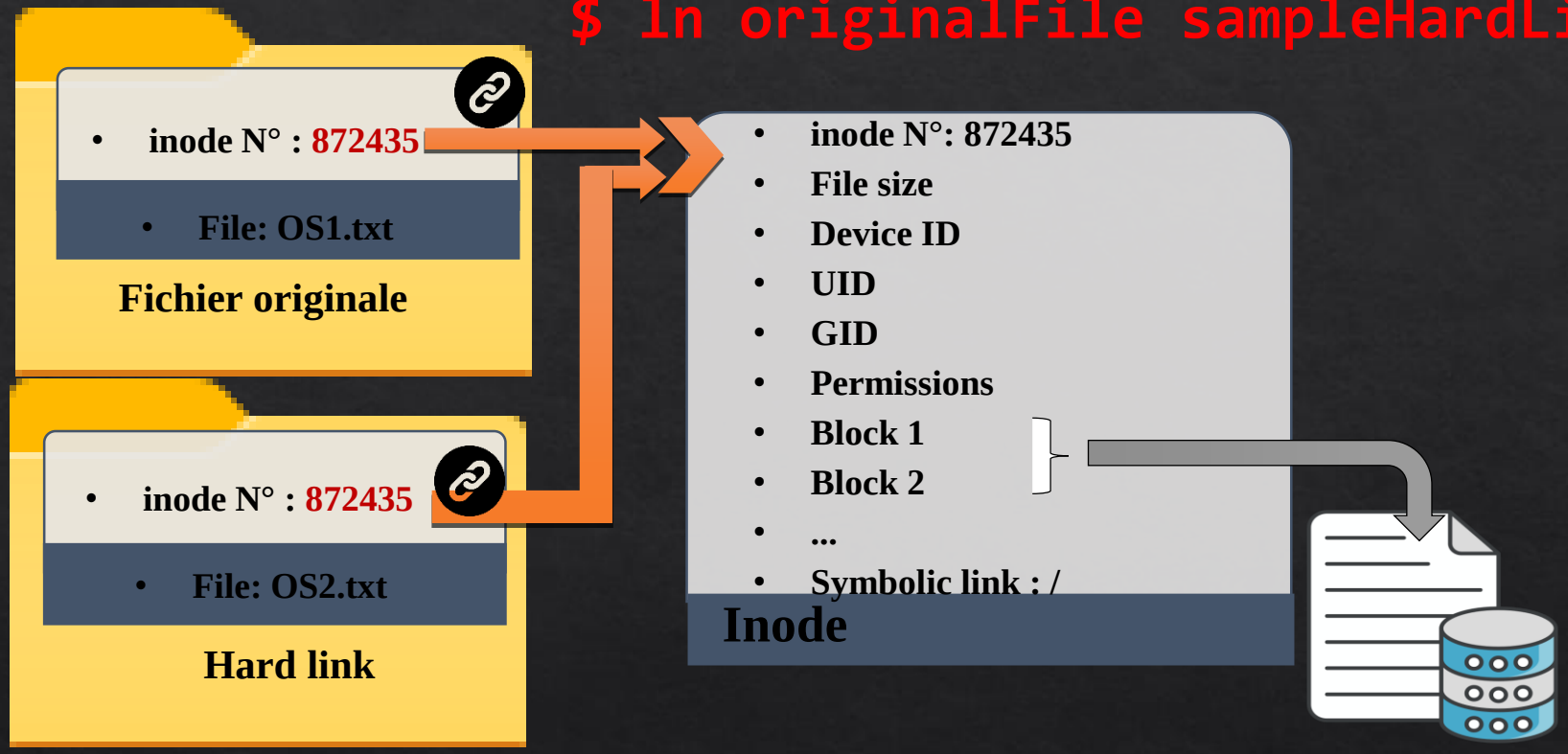
- A Linux filesystem has many hard links and symbolic links. A link is a connectivity between the filename and the actual data byte in the disk space. More than one filename can **link** to the same data.

- There are two types of links in Linux OS:
  1. **Hard Links**
  2. **Soft Links**

1. Hard Links

- They are the low-level links. It links more than one filename with the same Inode and it represents the physical location of a file.

```
$ ln originalFile sampleHardLink
```



When hard link is created for a file, it directly points to the Inode of the original file in the disk space, which means no new Inode is created. Directories are not created using hard links and they can not cross filesystem boundaries. When the source file is removed or moved, then hard links are not affected.

2. Soft Links (Symbolic Links)

- Soft links are very common. It represents a virtual or abstract location of the file. It is just like the shortcuts created in Windows. A soft link doesn't contain any information or content of the linked file, instead it has a pointer to the location of the linked file. In other words, a new file is created with new Inode, having a pointer to the Inode location of the original file.
- It is used to create link between directories and can cross filesystem boundaries. When the source file is removed or moved, then soft links are not updated.
- We'll study in deep about both the links how to create it and remove it.

```
$ ln -s originalFile sampleSoftLink
```

2. Soft Links (Symbolic Links)