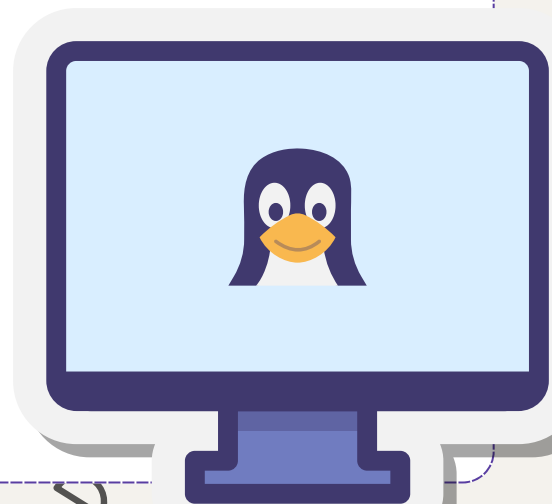


# HARD & SOFT LINK IN LINUX

*BY - Gauri Yadav*



# LIST OF TOPICS

01

WHAT IS LINKING IN LINUX?



02

TYPES OF LINK



03

WHAT IS INODE



04

WHAT IS HARD & SOFT LINK



05

HARD LINK V/S SOFT LINK



# What is linking in Linux?

- In Linux, a link is a reference to a file or directory. It allows multiple filenames to refer to the same file, providing flexibility in organizing and accessing data.
- Links enable efficient file management, reduce storage space usage, and facilitate the creation of symbolic relationships between files.
- Purpose and Benefits: Links serve several important purposes in Linux file systems:
- Efficient File Management: Links provide a means to organize files by creating multiple references to the same data. This reduces the need for duplicating files, saving storage space and ensuring consistency when managing related data.
- Storage Space Usage: By creating links instead of copying files, users can conserve disk space, especially when dealing with large files or when multiple instances of the same file are needed across different locations.
- Symbolic Relationships: Links enable the establishment of symbolic relationships between files or directories. This allows for the creation of aliases, shortcuts, or symbolic representations of files, making it easier to navigate and access data within the file system.



# Types of link?

Types of Links in Linux

Hard Link

Soft link



# What is Inode number?

- In a Unix-like operating system, including Linux, an "inode" (index node) is a data structure used to store information about a file or directory. Each file or directory in the file system is represented by an inode, which contains metadata about the file or directory, such as its permissions, ownership, timestamps, size, and pointers to the actual data blocks on the disk.
- Here's a detailed explanation of what an inode is and its components:
- Metadata: The inode stores various metadata attributes associated with the file or directory, including:
  - File type (regular file, directory, symbolic link, etc.).
  - Permissions (read, write, execute) for the owner, group, and others.
  - Owner and group ownership information.
  - Timestamps indicating when the file was last accessed, modified, or changed.
  - Size of the file in bytes.

- Pointers to data blocks containing the file's actual contents.
- Pointers to Data Blocks: For regular files, the inode contains pointers to the data blocks on the disk where the file's content is stored. These pointers may be direct pointers to data blocks for small files or indirect pointers (single, double, or triple) for larger files that cannot fit entirely within the inode.
- File System Pointer: In addition to data block pointers, the inode typically contains a pointer to the file system's data structure, such as the superblock or the inode table, allowing the file system to locate and manage the inode efficiently.
- Unique Identifier (Inode Number): Each inode in the file system is uniquely identified by an inode number, which serves as an index or identifier for the file or directory. Inode numbers are assigned sequentially when files or directories are created and remain constant throughout the lifetime of the file system object. They are used by the file system to quickly locate and access the corresponding inode and its associated data blocks.

# What is Hard & Soft link?

## Q1 What is hard link?

- When a hard link is created for a file, it essentially creates an additional directory entry (filename) that points directly to the same inode as the original file. This means that all hard links to the same file share the same set of data blocks on the disk.
- Unlike symbolic links (soft links), which reference the path to the target file, hard links directly reference the inode, making them indistinguishable from the original file in terms of data content.
- Since all hard links point to the same inode and share the same set of data blocks, any changes made to the file content, filename, or attributes through one hard link will affect all other hard links to the same file.
- For example, if you rename a file through one hard link, the change will be reflected in all other hard links to the same file. Similarly, changes to file permissions, ownership, or timestamps will be consistent across all hard links.
- This behavior ensures that all hard links maintain a synchronized state, providing a unified view of the file regardless of which hard link is accessed or modified.

## Q1 How to create Hard Link?

**Syntax:** *ln source\_file target\_link*

### ***Example 1: Creating a Hard Link:***

```
root@localhost:~  
[root@localhost ~]# touch testfile1.txt
```

*create new file*

```
[root@localhost ~]# ln testfile1.txt testfile2.txt  
[root@localhost ~]#
```

Source file

target file

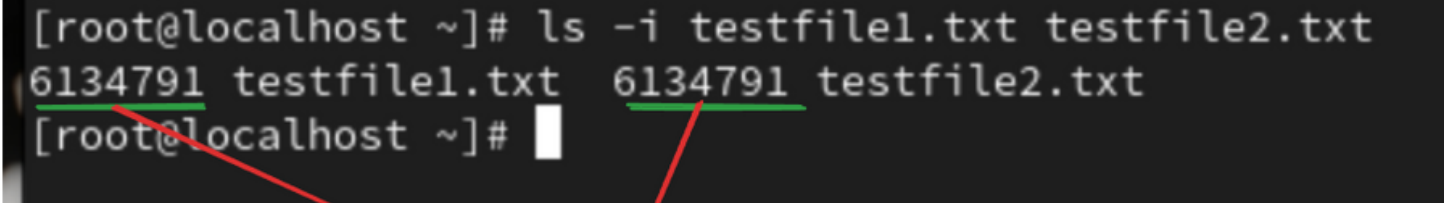
we created testfile1.txt but we didn't create testfile2.txt in this case  
testfile2.txt will be created on the spot and will be linked with testfile1.txt



## Q2 Observing Behavior with Hard Links and Inode Numbers:?

**command:** *ls -i testfile1.txt testfile2.txt*

```
[root@localhost ~]# ls -i testfile1.txt testfile2.txt
6134791 testfile1.txt 6134791 testfile2.txt
[root@localhost ~]#
```



Both the files shares the same Inode number

### Q3 What is Soft link?

- Unlike hard links, which are restricted to referencing files on the same filesystem, soft links have the flexibility to point to directories and files located on different filesystems.
- This characteristic makes symbolic links particularly useful for creating cross-filesystem references, allowing users to easily access files and directories residing in separate storage locations.
- When creating a symbolic link, the permissions of the target file or directory are retained. However, the symbolic link itself has its own permissions that determine who can read, write, or execute the link.
- This characteristic ensures that users accessing symbolic links have the appropriate permissions to access the target file or directory, subject to the permissions of the link itself.
- In file listings generated by commands like `ls -l`, symbolic links are distinguished from regular files and directories by a special indicator (typically an `l` character) in the file permissions field. This indicator signifies that the entry is a symbolic link rather than a regular file or directory.

## Q4 How to create Soft Link?

**Syntax:** `ln -s /path/to/source /path/to/symlink`

### *Example 1: Creating a Soft Link:*

```
# ln -s /home/user/documents/testfile1.txt /home/user/testfile2.txt
```

↓  
source

↓  
destination

## Q1 Observing Behavior with Soft Links?

Go to path and ls all the files



```
[root@localhost home]# ls -l /home/user  
total 0  
lrwxrwxrwx. 1 root root 34 Mar 22 17:31 testfile2.txt -> /home/user/documents/testfile1.txt  
[root@localhost home]#
```



This shows that both files are  
linked by soft link

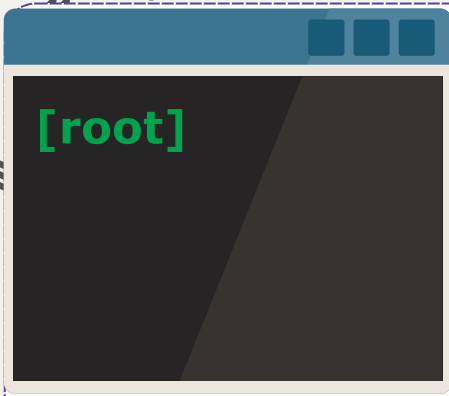
# Hard Link

## V/S

# Soft Link

Comparison Parameters	Hard link	Soft link
Inode number*	Files that are hard linked take the same inode number.	Files that are soft linked take a different inode number.
Directories	Hard links are not allowed for directories. (Only a superuser* can do it)	Soft links can be used for linking directories.
File system	It cannot be used across file systems.	It can be used across file systems.
Data	Data present in the original file will still be available in the hard links.	Soft links only point to the file name, it does not retain data of the file.
Original file's deletion	If the original file is removed, the link will still work as it accesses the data the original was having access to.	If the original file is removed, the link will not work as it doesn't access the original file's data.
Speed	Hard links are comparatively faster.	Soft links are comparatively slower.

cc: [GeeksforGeeks](#)



# THANK YOU

*Follow - @Gauri Yadav*

