Hard & Soft Link

An **inode** (short for "index node") is a data structure in Unix-like file systems, like Linux, that holds all the information about a file or directory **except for its name and the data itself**.

Here's a simpler breakdown:

- **Unique ID:** Every file has a unique inode number. Think of it as the file's social security number. The operating system uses this number to find the file's details.
- **Metadata:** The inode stores important information, or **metadata**, about the file, such as:
    - **Type:** What kind of file it is (e.g., a regular file, a folder).
    - **Permissions:** Who can read, write, or run the file.
    - **Owner:** Who created the file.
    - **Timestamps:** When it was created, last changed, or last opened.
    - **Size:** How big the file is.
    - **Location:** Pointers to where the actual file data is stored on the disk.
- **Separation:** This system separates a file's information from its content. This is a key design for how these file systems work.
- **File Limits:** A file system can only hold a certain number of inodes. If you run out of inodes, you can't create new files, even if you have plenty of disk space left.

This separation of the file name and the inode is what allows for hard links and symbolic links.

- **Hard Link:** Two or more file names can point to the **same inode**. They are essentially different names for the same file. If you delete one name, the file's data remains as long as at least one other hard link still points to the inode. The **link count** in the inode tracks how many names refer to it.
    - **Example:** You have a file named `report.txt`. You create a hard link to it called `final.txt`. Both names point to the same inode. If you delete `report.txt`, the file is still accessible through `final.txt`. Only when you delete `final.txt` as well does the link count drop to zero, and the system frees up the data blocks.
- **Symbolic (Soft) Link:** A symbolic link is a special file that contains the **path to another file name**. It has its own unique inode. Think of it as a shortcut. If you delete the original file, the symbolic link breaks because it's still trying to point to a name that no longer exists.
    - **Example:** You create a symbolic link named `daily_notes` that points to `~/Documents/notes/august/21.txt`. If you delete `21.txt`, the `daily_notes` link becomes useless, or "dangling," because it can no longer find its target.