CSCI 274 - Intro to Linux OS

Week 15 - Inodes and Linking

Max Gawason (maxgawason@mines.edu) [A]
Rocco Marchitto (rmarchitto@mines.edu) [B/C/D]

File System

An **inode (index node)** is a data structure on a traditional Unix-style file system. It stores basic information about a file, directory, or other file system object. Each object in the filesystem is associated with an inode.

Each and every file under Linux (and UNIX) has the following attributes (metadata) stored in an inode:

File type (executable, block special etc)	Group	File deletion time
Permissions (read, write etc)	File Size	Number of links (soft/hard)
Owner	File access, change and modification time	Access Control List (ACLs)

File System

Each inode stores the disk block location(s) of the object's data. This is the connection between the file within the filesystem and the actual data that it represents.

Each inode has a unique inode number.

You can use **ls -i** to see the inode number of a file

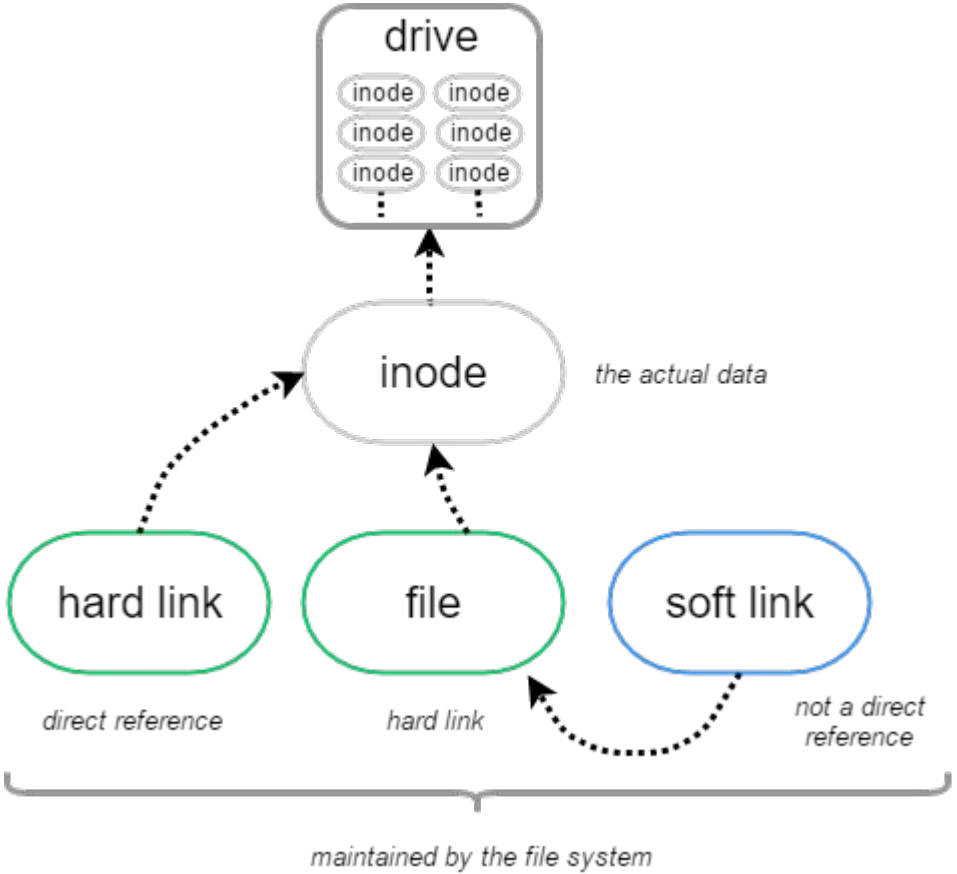
These numbers index into a table of inodes (maintained by the OS) in a known location on the device. From there, the kernel's file system driver can access the inode contents, including the location of the file - thus allowing access to the file’s data wherever it is stored.

File System

You can also use the **stat** command to see the inode number and attributes of a file.

Directories have inodes just like files.

There are two types of links: hard links and soft (symbolic) links.



Hard Link

With **hard links**, it is possible to associate multiple filesystem entities with a single inode. To create a hard link use the **ln** command as follows:

```
ln ./file1 ./file2
```

Here, file1 **must be an existing** filesystem entity. This will create file2 which will refer to the same inode as file1. This means that they both point to the same data. Editing file2 is the same as editing file1. Hard links **always refer to the inode** of the source, even if the source is moved or removed.

In other words, moving or removing file1 will not change the behavior of file2 at all.

Hard links **cannot** link directories.

Soft Link

Instead of linking to an inode, **soft links** refer to another link. We can create soft links by with **ln –s** as shown in the example below:

```
ln -s ./file1 ./file2
```

In the example above, file2 points to the link file1, which in turn points to the inode specifying the location of the data on disk. **Soft links** can link directories and cross filesystem boundaries (so a symbolic link to data on one drive or partition can exist on another drive or partition). However, if the source of the link is changed or removed, the soft link is not updated and will be **broken**.

Big summary

A **symbolic or soft link** is an actual link to the original file. If you delete the original file, the soft link has no value, because it points to a non-existent file.

Hard link is a mirror copy of the original file. If you delete the original file, the hard link will still has the data of the original file. Hard link acts as a mirror copy of the original file.