# Soft and Hard links in Unix/Linux

A link in UNIX is a pointer to a file. Like pointers in any programming languages, links in UNIX are pointers pointing to a file or a directory. Creating links is a kind of shortcuts to access a file. Links allow more than one file name to refer to the same file, elsewhere.

There are two types of links :

1. Soft Link or Symbolic links
2. Hard Links

These links behave differently when the source of the link (what is being linked to) is moved or removed. Symbolic links are not updated (they merely contain a string which is the pathname of its target); hard links always refer to the source, even if moved or removed.

For example, if we have a file a.txt. If we create a hard link to the file and then delete the file, we can still access the file using hard link. But if we create a soft link of the file and then delete the file, we can’t access the file through soft link and soft link becomes dangling. Basically hard link increases reference count of a location while soft links work as a shortcut (like in Windows)

1. Hard Links

* Each hard linked file is assigned the same Inode value as the original, therefore they reference the same physical file location. Hard links more flexible and remain linked even if the original or linked files are moved throughout the file system, although hard links are unable to cross different file systems.
* ls -l command shows all the links with the link column shows number of links.
* Links have actual file contents
* Removing any link, just reduces the link count, but doesn’t affect other links.
* We cannot create a hard link for a directory to avoid recursive loops.
* If original file is removed then the link will still show the content of the file.
* Command to create a hard link is:

$ ln [original filename] [link name]

2. Soft Links

* A soft link is similar to the file shortcut feature which is used in Windows Operating systems. Each soft linked file contains a separate Inode value that points to the original file. As similar to hard links, any changes to the data in either file is reflected in the other. Soft links can be linked across different file systems, although if the original file is deleted or moved, the soft linked file will not work correctly (called hanging link).
* ls -l command shows all links with first column value l? and the link points to original file.
* Soft Link contains the path for original file and not the contents.
* Removing soft link doesn’t affect anything but removing original file, the link becomes “dangling” link which points to nonexistent file.
* A soft link can link to a directory.
* Link across filesystems: If you want to link files across the filesystems, you can only use symlinks/soft links.
* Command to create a Soft link is:

$ ln -s [original filename] [link name]

# Difference Between Soft Link and Hard Link in Linux with Examples

Hard Link acts like a mirror copy of the original file. These links share the same inodes. Changes made to the original or hard linked file will reflect in the other. When you delete Hard Link nothing will happen to the other file. Hard links can't cross file systems.

Soft Link is an actual link to the original file. These Links will have a different Inodes value. Soft link points to the original file so if the original file is deleted then the soft link fails. If you delete the Soft Link, nothing will happen to file. The reason for this is, the actual file or directory’s inode is different from the "soft link" created file's inodes. Soft links can cross file systems.

## Hard links

Lets first create a "Test" directory and inside we create a new file "sample1".

$ mkdir Test

$ cd Test

$ touch sample1

Now, create a hard link to sample1. Name the hard link sample2.

$ ln sample1 sample2

Display inodes for both files using i argument of the ls command.

$ ls -il sample1 sample2

This is what you will get as output

1482256 -rw-r--r-- 2 bruno bruno 21 May 5 15:55 sample1

1482256 -rw-r--r-- 2 bruno bruno 21 May 5 15:55 sample2

From the output, you can notice that both sample1 and sample2 have the same inode number (1482256). Also, both files have the same file permissions and the same size.

Now Remove the original sample1

$ rm sample1

After removing Hard Link just have a look at the content of the "link" sample2.

$ cat sample2

You will still be able to see the contents of the file.

## Symbolic links

Let's create a soft link for the file sample2 using below command.

$ ln -s sample2 sample3

Display inodes for both using i argument of ls command.

$ ls -il sample2 sample3

This is what you'll get:

1482256 -rw-r--r-- 1 bruno bruno 21 May 5 15:55 FileB

1482226 lrwxrwxrwx 1 bruno bruno 5 May 5 16:22 FileC -> FileB

From the output, you can notice that the inodes are different and the symbolic link has an "l" before the rwxrwxrwx. The permissions are different for the link and the original file because it is just a symbolic link.

Now list the contents:

$ cat sample2

$ cat sample3

Now remove the original file

$ rm sample2

And then check the Test directory:

$ ls

It will still display symbolic link sample3 but if you try to list the contents, it will tell you that there is no such file or directory.

$ cat sample3

Now you know about some of the key differences between Hard Links and Soft Links that will make it easier for you to access files and run programs.