There are two types of links available in Linux — **Soft Link and Hard Link**. Linux ln command is used to create either soft or hard links.

This article explains how to create soft link, how to create hard link, and various link tips and tricks with 10 practical examples.

```
$ ls -l

total 4

lrwxrwxrwx 1 chris chris 10 2010-09-17 23:40 file1 -> sample.txt

-rw-r--r-- 1 chris chris 22 2010-09-17 23:36 sample.txt
```

The 1st character in each and every line of the ls command output indicates one of the following file types. If the 1st character is l (lower case L), then it is a link file.

- **–** regular file
- **l** link file
- **d** directory
- **p** pipe
- **c** character special device
- **b** block special device

1. What is Soft Link and Hard Link?

Soft Link
Linux OS recognizes the data part of this special file as a reference to another file path. The data in the original file can be accessed through the special file, which is called as Soft Link.

To create a soft link, do the following (ln command with -s option):

```
$ ln -s /full/path/of/original/file /full/path/of/soft/link/file
```

Hard Link
With Hard Link, more than one file name reference the same inode number. Once you create a directory, you would see the hidden directories "." and ".." . In this, "." directory is hard linked to the current directory and the ".." is hard linked to the parent directory.

> When you use link files, it helps us to reduce the disk space by having single copy of the original file and ease the administration tasks as the modification in original file reflects in other places.

To create a hard link, do the following (ln command with no option):

```
$ ln /full/path/of/original/file /full/path/of/hard/link/file
```

2. Create Symbolic Link for File or Directory

Create a symbolic link for a File
The following examples creates a symbolic link library.so under /home/chris/lib, based on the library.so located under /home/chris/src/ directory.

```
$ cd /home/chris/lib



$ ln -s /home/chris/src/library.so library.so



$ ls -l library.so

lrwxrwxrwx  1 chris chris       21 2010-09-18 07:23 library.so ->
/home/chris/src/library.so
```

Create a symbolic link for a Directory
Just like file, you can create symbolic link for directories as shown below.

```
$ mkdir /home/chris/obj



$ cd tmp



$ ln -s /home/chris/obj objects



$ ls -l objects

lrwxrwxrwx 1 chris chris       6 2010-09-19 16:48 objects -> /home/chris/obj
```

Note: The inode of the original file/directory and the soft link should not be identical.

3. Create Hard Link for Files

The inode number for the hard linked files would be same. The hard link for files can be created as follows,

```
$ ln src_original.txt dst_link.txt



$ ls -i dst_link.txt

253564 dst_link.txt



$ ls -i src_original.txt

253564 src_original.txt
```

Note: Unix / Linux will not allow any user (even root) to create hard link for a directory.

4. Create Links Across Different Partitions

When you want to create the link across partitions, you are allowed to create only the symbolic links. Creating hard link across partitions is not allowed, as Unix can't create/maintain same inode numbers across partitions.

You would see the **"Invalid cross-device link"** error when you are trying to create a hard link file across partitions.

```
# mount /dev/sda5 /mnt



# cd /mnt



# ls
```

```
main.c Makefile



# ln Makefile /tmp/Makefile

ln: creating hard link `/tmp/Makefile' to `Makefile': Invalid cross-device link
```

And the symbolic link can be created in the same way as we did in the above.

5. Backup the Target Files If it Already Exists

When you create a new link (if another file exist already with the same name as the new link name), you can instruct ln command to take a backup of the original file before creating the new link using the –backup option as shown below.

```
$ ls

ex1.c  ex2.c



$ ln --backup -s ex1.c ex2.c



$ ls -lrt

total 8

-rw-r--r-- 1 chris chris 20 2010-09-19 16:57 ex1.c

-rw-r--r-- 1 chris chris 20 2010-09-19 16:57 ex2.c~

lrwxrwxrwx 1 chris chris  5 2010-09-19 17:02 ex2.c -> ex1.c
```

Note: If you don’t want the backup and overwrite the existing file then use -f option.

6. Create Link Using “No-Deference” ln Command Option

While creating a new soft link, normally OS would de-reference the destination path before it creates the new soft link.

Sometimes you might not want ln command to create the new link, if the destination path is already a symbolic link that is pointing to a directory.

Following examples shows a normal way of creating soft link inside a directory.

```
$ cd ~



$ mkdir example



$ ln -s /etc/passwd example



$ cd example/



$ ls -l

total 0

lrwxrwxrwx 1 root root 16 2010-09-19 17:24 passwd -> /etc/passwd
```

In case the “example” directory in the above code-snippet is a symbolic link pointing to some other directory (for example second-dir), the ln command shown will still create the link under second-dir. If you don’t want that to happen, use ln -n option as shown below.

```
$ cd ~



$ rm -rf example



$ mkdir second-dir
$ ln -s second-dir example



$ ln -n -s /etc/passwd example

ln: creating symbolic link `example': File exists
```

Note: In the above example, if you don’t use the -n option, the link will be created under ~/second-dir directory.

7. Create Link for Multiple Files at the Same Time

In the following example, there are two directories — first-dir and second-dir. The directory first-dir contains couple of C program files. If you want to create soft links for these files in second-dir, you’ll typically do it one by one. Instead, you can create soft list for multiple files together using -t option as shown below.

```
$ ls

first-dir second-dir



$ ls first-dir

ex1.c  ex2.c



$ cd second-dir



$ ln -s ../first-dir/*.c -t .



$ ls -l

total 0

lrwxrwxrwx 1 chris chris 14 2010-09-19 15:20 ex1.c -> ../first-dir/ex1.c
lrwxrwxrwx 1 chris chris 14 2010-09-19 15:20 ex2.c -> ../first-dir/ex2.c
```

Keep in mind that whenever you are creating link files with -t option, it is better to go into target directory and perform the link creation process. Otherwise, you would face the broken link files as shown below.

```
$ cd first-dir



$ ln -s *.c /home/chris/second-dir



$ cd /home/chris/second-dir

$ ls -l

total 0

lrwxrwxrwx 1 chris chris 5 2010-09-19 15:26 ex1.c -> ex1.c
lrwxrwxrwx 1 chris chris 5 2010-09-19 15:26 ex2.c -> ex2.c
```

Instead, you might also use actual path for source files to create the link properly.

8. Removing the Original File When a Soft Link is pointing to it

When the original file referred by a soft-link is deleted, the soft link will be broken as shown below.

```
$ ln -s file.txt /tmp/link



$ ls -l /tmp/link

lrwxrwxrwx 1 chris chris 9 2010-09-19 15:38 /tmp/link -> file1.txt



$ rm file.txt
$ ls -l /tmp/link

lrwxrwxrwx 1 chris chris 9 2010-09-19 15:38 /tmp/link -> file1.txt
```

9. Links Help You to Increase the Partition Size Virtually

Let us assume that you have two partitions – 5GB and 20GB. The first partition does not have too much free space available in it. If a program located on the first partition needs more space (For example, for it's log file), you can use some of the space from the second partition by creating a link for the log files as shown below.

Consider that partition1 is mounted on /, and partition2 is mounted to /mnt/. Let us assume that the logs that are located on partition1 is running out of space, and you've decided to move them to partition2. You can achieve this as shown below.

```
$ mkdir /mnt/logs



$ cd /logs



$ mv * /mnt/logs



$ cd /; rmdir logs



$ ln -s /mnt/logs logs
```

10. Removing the Hard Linked Files

When you delete a file that is hard linked, you would be still able to access the content of the file until you have the last file which is hard linked to it, as shown in the example below.

Create a sample file.

```
$ vim src_original.txt
```

```
Created this file to test the hard link.
```

Create a hard link to the sample file.

```
$ ln src_original.txt dst_link.txt
```

Delete the original file.

```
$ rm src_original.txt
```

You can still access the original file content by using the hard link you created.

```
$ cat dst_link.txt

Created this file to test the hard link.
```