**\*8. 1 Links - Hard links Soft links**

**8.1.1 Hard links**

----------

Hard links are physical copies of files. Let's say you create a new

file named file, so rm

**touch file**

Now **ls -l file**

will show the long listing of that file, for example:

***-rw-r--r-- 1 mark.fernandes users 0 Oct 27 11:05 file***

Notice the link number is 1 (after the permissions). To check the

***inode number*** of file use:

**ls -il file**

2156060563 -rw-r--r-- 1 mark.fernandes users 0 Oct 27 11:05 file

The inode number is **2156060563** for file. Now to **create a hard link** use:

**ln file hlink**

To check that a hard link was created, use **ls -il file hlink**

2156060563 -rw-r--r-- 2 mark.fernandes users 0 Oct 27 11:05 file

2156060563 -rw-r--r-- 2 mark.fernandes users 0 Oct 27 11:05 hlink

Notice **the i-node number** (first number in the long list) **is the same**

and that the link count has gone up by **1**. This means ***any changes to***

***file automatically get copied to hlink***. Try

**echo 'Hello' > file**

To see the contents use **cat file**

****

Now try **cat hlink**



Notice the content for both are the same. To delete a hard link use

**rm hlink**

Now run

**ls -il file**

You'll see

2156060563 -rw-r--r-- **1** mark.fernandes users 6 Oct 27 11:05 file

Notice the link counter has gone back down to **1.** The number 6 is the number of characters in file.

***\* Hard links can only be made to files, not to directories***

***\* Every new link created increases the link counter by 1 but does not reduce***

disk space by the amount used in the file. So if the file was 1GB in

size, a hard link \*does not\* reduce the available disk space by 1GB.

\* **Hard links create 'live' backups to files:**

- if content of original file changes then backup file changes too

- **ls -l** shows the number of hard links to a file,

- increases by one each time a hard link is added with **ln**

- decreases by one each time a hard link file is removed by **rm**

\* Another example explaining hard links

**$ touch hello** # create empty file hello

**$ echo "Hello World" >> hello** # add some content to it

**$ ln hello hello.backup** # create hard link called hello.backup

**$ ls -l** # note that link number shows 2 (up by 1)

**$ rm hello** # remove original file

**$ cat hello.backup** # hard link has content from original file

**8.1.2 Soft links**

Soft links are shortcuts to a file or directory. To create a soft link use:

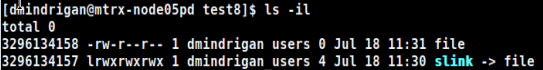
**ln -s file slink**

Now

**ls -l file slink** will show

-rw-r--r-- 1 mark.fernandes users 6 Oct 27 11:14 file

lrwxrwxrwx 1 mark.fernandes users 4 Oct 27 11:21 slink -> file

and

**ls -il file slink**

2156060563 -rw-r--r-- 1 mark.fernandes users 6 Oct 27 11:14 file

2156060564 lrwxrwxrwx 1 mark.fernandes users 4 Oct 27 11:21 slink -> file

Notice the inode numbers are different. If you tried

**cat slink**

You will see the contents of file, which would be 'Hello'

\* Soft links can be made to files and directories

\* Deleting the target does not delete the link (but the link is now

useless).

So  **rm file**

And **ls –il** will show 

lrwxrwxrwx 1 mark.fernandes users 4 Oct 27 11:21 slink -> file

but the link is pointing nowhere, if you try

**cat slink**

You will get an error message, like so:



**Hard link vs. Soft link in Linux or UNIX**

* Hard links cannot link directories.
* Cannot cross file system boundaries.

Soft or symbolic links are just like hard links. It allows associating multiple filenames with a single file. However, symbolic links allows:

* To create links between directories.
* Can cross file system boundaries.

These links behave differently when the source of the link is moved or removed.

Symbolic links are not updated.

Hard links always refer to the source, even if moved or removed.

Practice

---------

1. Create a rough\_work directory.

2. In the rough\_work directory, attempt to create the following:

a. soft link to file /usr/share/dict/words; call the soft link words

b. hard link to file /usr/share/dict/words; call the hard link words2

c. soft link to dir /usr/share/dict; call the soft link dict

d. hard link to dir /usr/share/dict; call the hard link dict2

What works and what does not work and why?

3. Using relative pathnames, create soft link to file 'words' in /usr/share/dict

called words3

4. Change directory so you are no longer in rough\_work and try (note:

path/to/ is not a real path but will change based on directory you've

changed to and where rough\_work is located relative to your pwd)

a. head -2 path/to/words

b. head -2 path/to/words2

c. head -2 path/to/words3

What works and what does not work and why

5. Make a directory called temp inside rough\_work, but keep your pwd

as rough\_work. Using relative pathnames, create a soft link, in

directory temp, called words4. words4 links to file 'words' in

/usr/share/dict. Next try

a. head -2 path/to/words4

Change pwd into rough\_work/temp and try

b. head -2 path/to/words4

What works and what does not work and why

Practice (soft links)

---------------------

1. Make the following directory tree and answer this question using

relative pathnames: Your pwd is school. Create a soft link, to

backup directory, in documents called backup. In other words, you

are asked to do the following:

- (your PWD is) school

- (make soft link to) backup

- (soft link is in) documents

- (link name is ) backup

rough\_work/

├── backup

│   └── documents.tar.gz

├── documents <= make soft link 'backup' here to directory backup

│   ├── private

│   │   └── finances.csv

│   ├── quote.txt

│   ├── school <= You are here

│   │   └── uli101

│   └── summary.odp

└── tmp

└── xyz.tmp

based on PWD of school, your answer on the command line is

ln -s ../backup ../backup

or

ln -s ../backup ..

(soft) (target) (location relative to pwd with link name)

2. pwd is school. Create soft link called temp, in backup, to tmp. Use

relative pathnames.