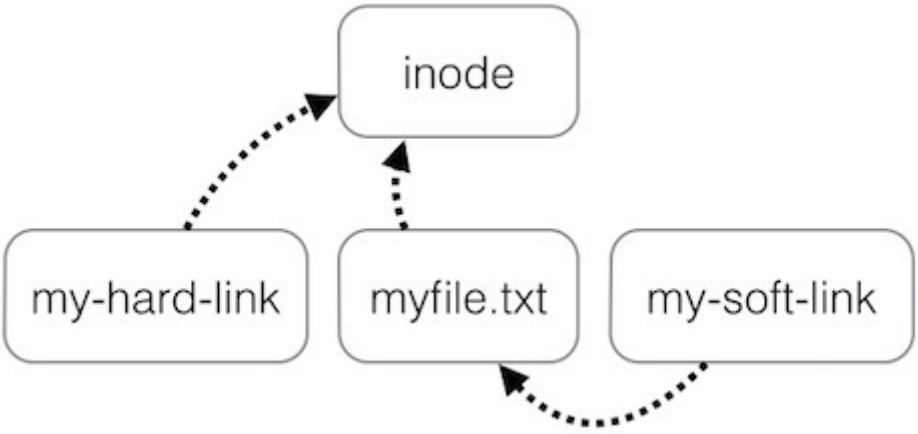
Andrew

7 Followers · About Follow

Hard links and Symbolic links — A comparison

Andrew Jan 16, 2018 · 4 min read



Visualized path difference between hard link and symbolic link references

Hard links and symbolic links are two different methods to refer to a file in the hard drive. These methods are part of the filesystem that organizes what file is what and where. A hard link is essentially a synced carbon copy of a file that refers directly to the inode of a file. Symbolic links on the other hand refer directly to the file which refers to the inode, a shortcut. In order to understand how symbolic and hard links work, we will need to go over what are inodes.

What is an inode?

The inode is a database that describes the file/directory attributes such as metadata and the physical location on the hard drive. They are essentially the numerical equivalent of a full address. With an inode, the OS can retrieve information about the

be moved to a different location on th
it automatically. This will be importan

What is a hard link?

A hard link is a direct reference to a fil
and not directories. By using a hardlin
location and the hardlink will still poi
pointing to that file. There is no refere
can only refer to files within the same volume otherwise symbolic links will be needed. To make a hard link of a file, you will require the ln command and refer to the source file before naming what the hard link will be named. Here is an example of how a hard link named test 2 will be made.

```
vagrant@vagrant-ubuntu-trusty-64:~/holberton-system_engineering-devops/0x03-s
hell_variables_expansions/test$ touch test
vagrant@vagrant-ubuntu-trusty-64:~/holberton-system_engineering-devops/0x03-s
hell_variables_expansions/test$ ls
aaa.jpg  aa.jpg  a.jpg  reddit.html  test
vagrant@vagrant-ubuntu-trusty-64:~/holberton-system_engineering-devops/0x03-s
hell_variables_expansions/test$ ln test test2
vagrant@vagrant-ubuntu-trusty-64:~/holberton-system_engineering-devops/0x03-s
hell_variables_expansions/test$ ls
aaa.jpg  aa.jpg  a.jpg  reddit.html  test  test2
vagrant@vagrant-ubuntu-trusty-64:~/holberton-system_engineering-devops/0x03-s
hell_variables_expansions/test$
```

First made the test file before making hard link test2

The file test should be completely empty and I will add “Hello” to it via the hard link.

```
vagrant@vagrant-ubuntu-trusty-64:~/holberton-system_engineering-devops/0x03-s
hell_variables_expansions/test$ emacs test2
```

Typing in Hello into the file test via test2

```
Hello
test2 (END)
```

Opening test 2 with Hello in it

```
Hello
test (END)
```

As seen in the photos above, I have ch
"Hello". By opening the original file, t
make sure the files are referring to the

```
vagrant@vagrant-ubuntu-trusty-64:~
hell_variables_expansions/test$ ls
275215 aaa.jpg  275179 a.jpg
275182 aa.jpg   275207 reddit.html
```

test and test2

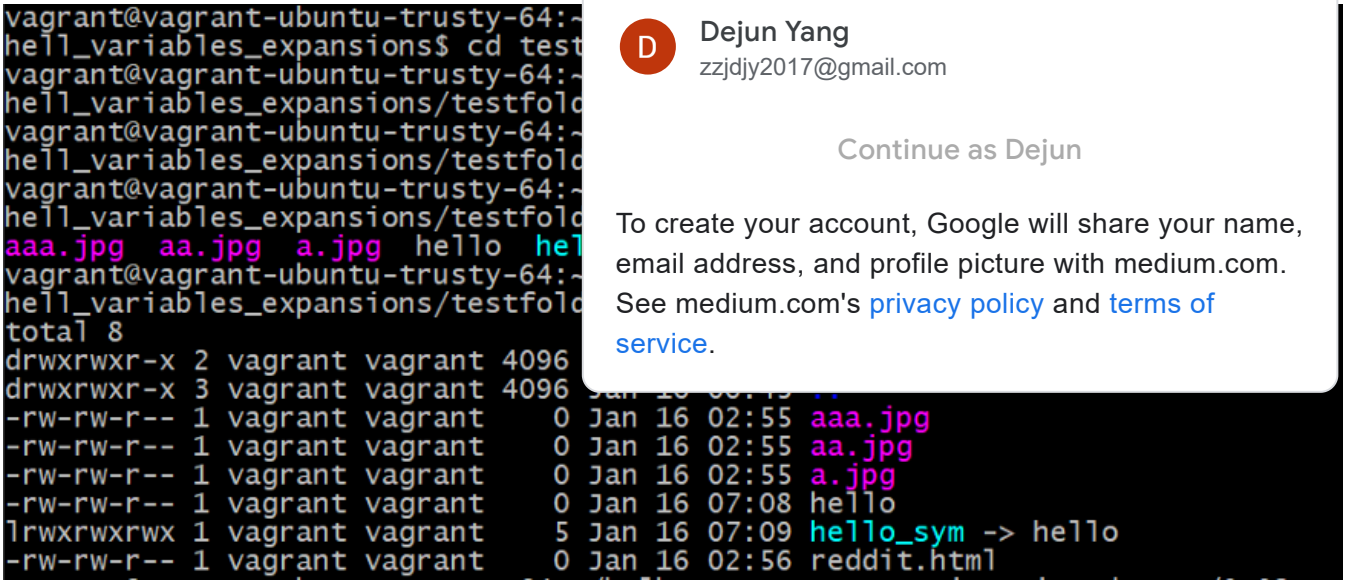
Now what will happen if we copy over a similar file called test from a different folder into this folder? For this experiment we will change the folder name from 'test' to 'test folder'.

```
vagrant@vagrant-ubuntu-trusty-64:~/holberton-system_engineering-devops/0x03-s
hell_variables_expansions$ cp test ./testfolder/test
vagrant@vagrant-ubuntu-trusty-64:~/holberton-system_engineering-devops/0x03-s
hell_variables_expansions$ cd testfolder/
vagrant@vagrant-ubuntu-trusty-64:~/holberton-system_engineering-devops/0x03-s
hell_variables_expansions/testfolder$ emacs test
vagrant@vagrant-ubuntu-trusty-64:~/holberton-system_engineering-devops/0x03-s
hell_variables_expansions/testfolder$ ls -i
275215 aaa.jpg  275179 a.jpg        275219 test   275219 test2~
275182 aa.jpg   275207 reddit.html  275219 test2
vagrant@vagrant-ubuntu-trusty-64:~/holberton-system_engineering-devops/0x03-s
hell_variables_expansions/testfolder$ emacs test2
vagrant@vagrant-ubuntu-trusty-64:~/holberton-system_engineering-devops/0x03-s
hell_variables_expansions/testfolder$ cd ..
vagrant@vagrant-ubuntu-trusty-64:~/holberton-system_engineering-devops/0x03-s
hell_variables_expansions$ mv ./test ./testfolder/test
vagrant@vagrant-ubuntu-trusty-64:~/holberton-system_engineering-devops/0x03-s
hell_variables_expansions$ cd testfolder/
vagrant@vagrant-ubuntu-trusty-64:~/holberton-system_engineering-devops/0x03-s
hell_variables_expansions/testfolder$ ls -i
275215 aaa.jpg  275179 a.jpg        275220 test   275219 test2~
275182 aa.jpg   275207 reddit.html  275219 test2
```

Here we can see that the cp command does not change the inode value of the original value but mv does. We have copied over a file from the parent directory into 'testfolder' and the inode value has not changed. It is only when you move over a file and replace the file that the inode value changes.

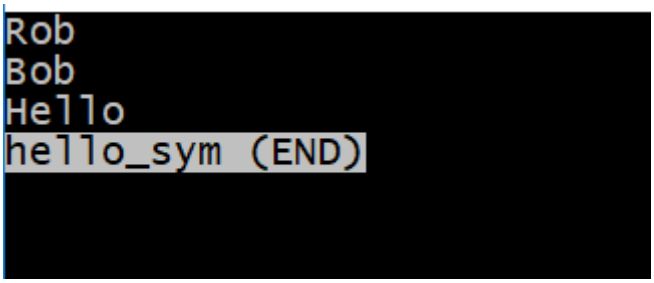
What are symbolic links?

Symbolic links are essentially shortcuts that reference to a file instead of its inode value. This method can be applied to directories and can reference across different hard disks/volumes. Since the symbolic link is referring to the original file and not its

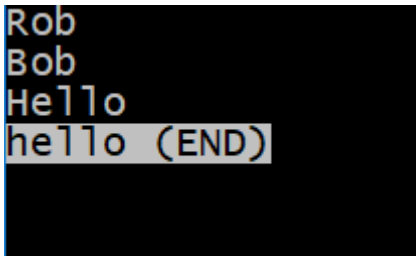


Making a symbolic link. Note the link has an arrow pointing to the original file in its filename

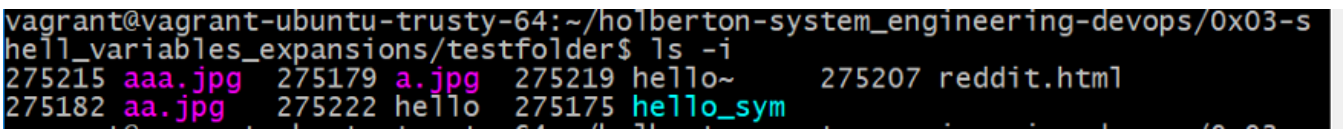
Since the symbolic link is a link that directs to the original file, changing the symboliclink should change the original file.



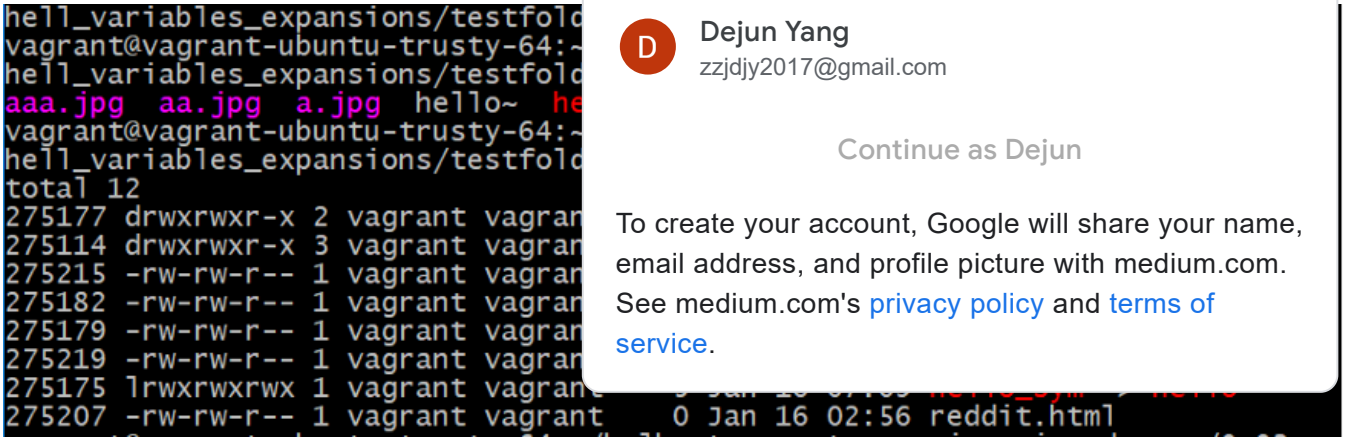
Changed the link hello_sym



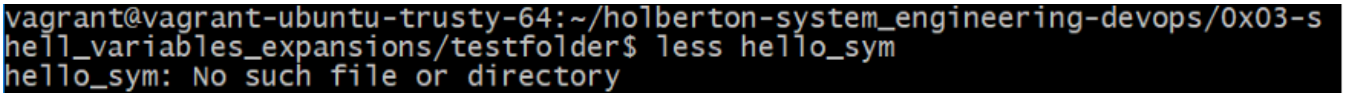
The change is reflected in the original file



A quick ls -i check shows they have different inodes

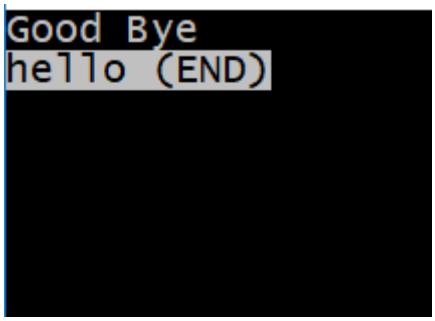


Moving the original file to a different folder broke the link

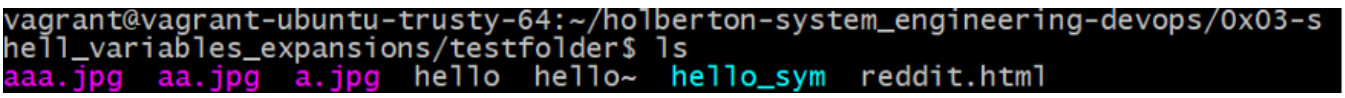


Opening the link shows the link is broken

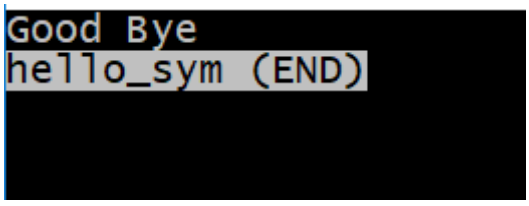
So symbolic links can be seen as a static link to the last known location of the original file. The link should work even if you replace the original file with a different file with the same name.

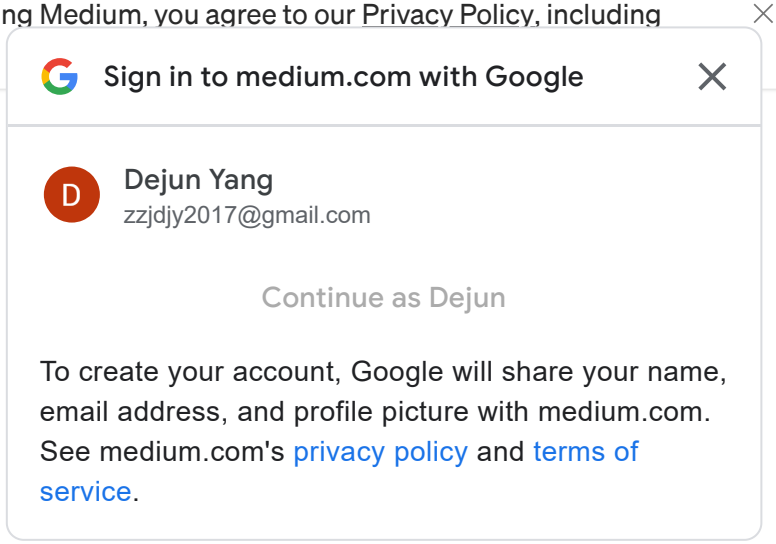


Made a new file hello with new contents



The link is now working again





Web Development Inode Hard Link