How to manage symlinks in Linux

Symlink (symbolic links) are files that link directly to other files. In Linux systems, symlinks act as shortcuts.

Symlink (symbolic links) are files that link directly to other files. In Linux systems, symlinks act as shortcuts.

The term **'symlink'** is a combination of two **'symbolic'** and **'link'** terms, highlighting the usefulness of these files as a reference for other things.

The so-called hard link functions as copies of the files they refer to, rather than literally links. Soft links or symbolic links are simply aimed at their target. Deleting these links has no effect on the files they point to and many necessary symbolic links can be created for ease of use. This makes them a great option to point to files on the same file systems and partitions.

Instructions on symlink management in Linux systems

1. Uses of symlinks
2. Problems with symbolic links
3. Managing symbolic links
4. Create symbolic link
5. Find symbolic links
6. Fix symbolic links

Uses of symlinks

It is very important to change the basic structure of a file system for a single application to run more efficiently. Instead, symlinks are often used to make the problem simpler and create artificial file hierarchies for programs to reference, without affecting the location of the original files.

Using symbolic links makes it easier to regulate different programs, but it also complicates the file system analysis.

Problems with symbolic links

When symbolic links are working properly, they clearly explain the path to a file that actually exists. However, problematic symlink will point to files that do not exist or have been deleted. These symbolic links create confusion for both users and programs depending on their accuracy.

If the files are targeted by symbolic links that are swapped, the link will continue to point to the new file, ignoring its entire contents. This 'blind faith' can create link chaining and relative links.

Specifically, linking links can lead to periodic links (infinite circles of links), if a link references the second link, then turns back to the first link. .

Such hard work is not necessarily due to human intervention. Non-optimal device installation standards and certain automated processes can contribute to the creation of what is called a 'dead link'. This is exactly where management techniques will work.

Managing symbolic links

In Linux systems, there are many built-in utilities to handle symbolic links. A default inclusion in coreutils is **ln,** facilitating the creation of such links from the terminal.

However, to fully manage symbolic links, you will need to be able to find and analyze them quickly. A simple command line option worth considering for this purpose is named **Symlinks.**

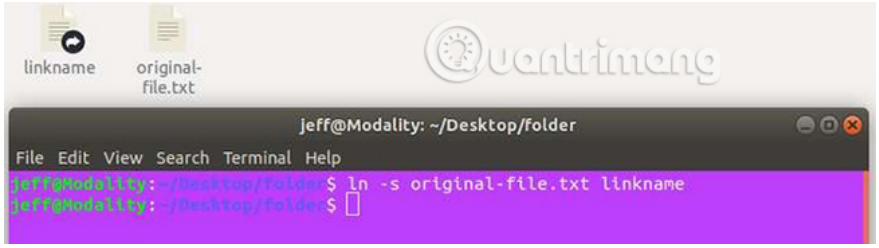
Although some Linux distributions (such as Fedora) that come with this tool are installed by default, other distros, such as Ubuntu, are not. To install Symlinks in Ubuntu, simply open a terminal window and enter the following information:

```
sudo apt install symlinks
```

Create symbolic link

Creating symbolic links from the terminal in Linux is easy. Enter the following code, change the **'original-file.txt' to** the name and file extension of the target you selected, then change the **'linkname'** to whatever you want.

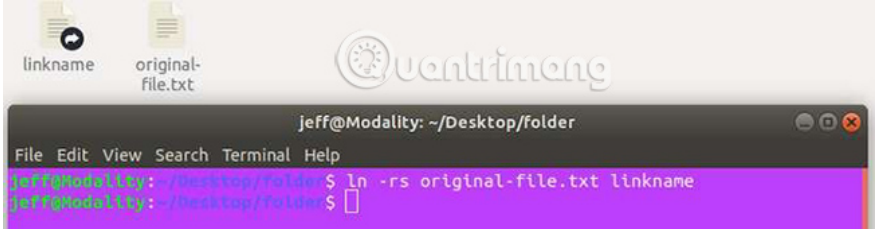
```
ln -s original-file.txt linkname
```



The **ln** utility is to create a link and it will do so when you run it. The **-s** symbol included in the above command makes the link created symbolically.

Relative symbolic links can also be created by adding a **-r** symbol to the same command:

```
ln -rs original-file.txt linkname
```



Relative links still work regardless of changes in mount points.

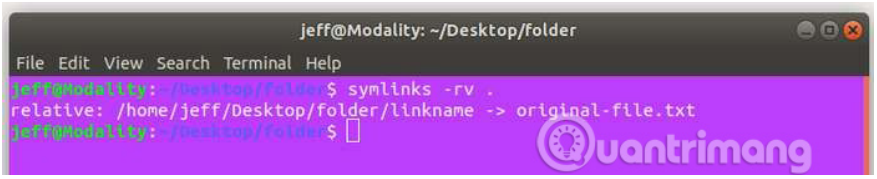
Find symbolic links

The Symlinks utility mentioned above gives us a simple way to find symbolic links in a certain directory. The command to find symbolic links is as follows: (Change **'directory-name'** to the full path to the directory you want to search).

```
symlinks -v directory-name
```

Adding **r** to this command will let Symlinks recursively check the files in the specified directory. The command will look like this:

```
symlinks -rv directory-name
```



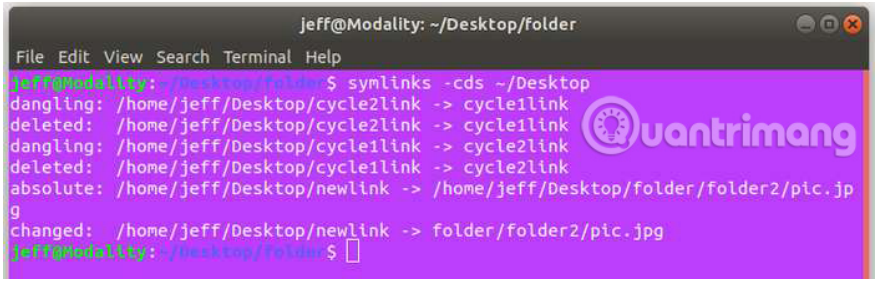
Be careful to use recursion if you are worried about problems with cyclic links. Cyclic links are self-repeating links. They can cause Symlinks to hang when it tries to repeat their infinite structure.

The non-recursive version of the command shown above will simply reveal any existing cyclic links that could be 'suspended', or otherwise broken. Symlinks tool has the ability to edit broken links like that.

Fix symbolic links

Modifying symbolic links in a given folder is relatively simple with Symlinks. Here is the command to use:

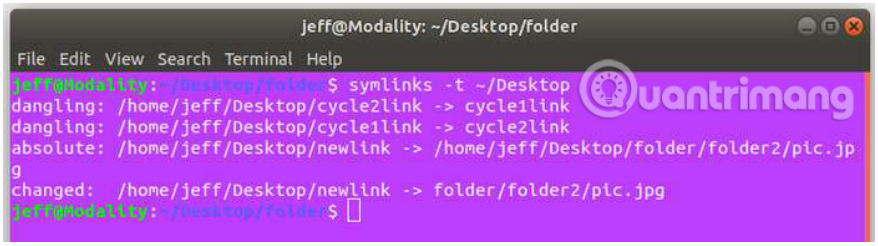
```
symlinks -cds directory-name
```



The above command does many things at once. It converts any absolute link that it finds into relative links, eliminates hanging links and shortens long links (links with lots of '**./**' in the path). .

If you are unsure of the potential result of running this command, you can check what a **-t** will do, without changing any details by running the following command:

```
symlinks -t directory-name
```



Hopefully, now you have a better understanding of what symbolic links are and how you can manage them effectively. Check out the rest of Symlinks utility to perform more specific operations on your file system by visiting the following link:

```
http://manpages.ubuntu.com/manpages/trusty/man1/symlinks.1.html
```

Hope you are succesful.