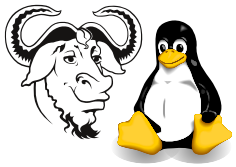

Links: Hard Links and Symbolic Links — Solutions

1 Aim

You will understand hard and soft links to the point where you can use them easily to solve many problems.

2 Background

A *link* gives another name to access a file by.

Links (particularly symbolic links) are incredibly useful in many situations for the system administrator. They can solve problems such as installing software when the partition you must install it in doesn't have enough space, although another partition does have enough space. There are very many other applications.

2.1 Some Definitions

A *directory* is a file that is has two entries for each file: a *file name* and an *inode number*. The inode number uniquely identifies a data structure that contains all the information about the file except its file name. To see the directory entries as they are stored in the directory, type:

`$ ls -Ui` ⟨*directory-name*⟩

A *hard link* is another directory entry for a file that has another directory entry in the same partition. Each of these links has the same inode number. If the links are in the same directory, they must have different file names.

To create a hard link ***`link`*** to the file ***`source`***, type `ln` ***`source link`***:

```
$ ln source link
$ ls -li
total 8
 842707 -rw-r--r--    2 nicku    nicku        2681 Dec 16 20:46 link
 842707 -rw-r--r--    2 nicku    nicku        2681 Dec 16 20:46 source
```

Note that the source and the link both have the same inode number. Both links are equally "the file".

A *symbolic link* or *soft link* is a small file that contains the name of another file. When you refer to the link, the action takes place on the file whose name it contains. We'll discuss symbolic links in detail in section 4.

3 Limitations of Hard and Soft Links

3.1 Hard Links

- A hard link must be to a file that is not a directory and must be within the same disk partition.
- The source file must exist.
- It is hard to find all the links to the file if they are scattered in different places on the partition.

3.2 Symbolic Links

A symbolic link has few limitations, but there are some; the source file must be visible from the position of the link. Some applications such as the ftp server have a different root directory from the main directory system (using the *chroot* system call). Another case is where a partition is mounted in a different place from usual. In such cases you need to use a *relative* symbolic link. A relative symbolic link is linked to a name that does not start with a `/`. In general, it is often safer to use relative rather than absolute symbolic links.

4 All About Symbolic Links

To create a symbolic link, you type:

`$ ln -s` ⟨*source*⟩ ⟨*link*⟩

The result will look something like this:

```
$ ln -s source link
$ ls -li source link
 858645 lrwxrwxrwx    1 nicku    nicku           6 Dec 16 20:47 link -> source
 858644 -rw-r--r--    1 nicku    nicku        2681 Dec 16 20:46 source
```

The size of this link is 6, the number of characters in the name 'source'. The size of the link is equal to the number of characters in the file that the link links to. This is a *relative symbolic link* because the name `source` does not begin with a `/`.

An important thing to understand is that the content of the symbolic link (the name that it is linked to) is exactly what you type as the first name in

`$ ln -s` ⟨*source*⟩ ⟨*link*⟩

4.1 Checking what file a link points to

It is easy to check what file a link points to with the `-L` option to `ls`:

```
$ ls -Lil source link
 858644 -rw-r--r--    1 nicku    nicku        2681 Dec 16 20:46 link
 858644 -rw-r--r--    1 nicku    nicku        2681 Dec 16 20:46 source
```

Dangling links are links that point to a file that does not exist:

```
$ ln -s foo bar
$ ls -l foo bar
ls: foo: No such file or directory
lrwxrwxrwx    1 nicku    nicku           3 Dec 16 20:59 bar -> foo
```

The `ls` program colours bad links white on a red background like **this**. Good links are coloured light-blue like this.

4.2 Making a link in another directory.

Some people get confused when making a link to a file in another directory. Here I am, in the `/bin` directory and I want to make a link to the `ls` program in my `~/bin` directory:

```
1055 $ pwd
/bin
$ ln -s ls ~/bin
$ ls -l ~/bin/ls
lrwxrwxrwx   1 nicku  nicku   2 Dec 16 21:53 /home/nicku/bin/ls -> ls
```

Oh dear, I got a dangling link! How to get it right? The thing to realise is that whatever you type as the *source* in

`ln -s` *source link*

is what will appear on the right side of arrow `->` when you list the link with `ls -l`. This is the right way to do it:

```
$ pwd
/bin
$ ln -s /bin/ls ~/bin
$ ls -l ~/bin/ls
lrwxrwxrwx   1 nicku  nicku   7 Dec 16 22:05 /home/nicku/bin/ls -> /bin/ls
```

Or we could make a *relative link* (see section 3.2) like this:

```
$ pwd
/bin
$ ln -s ../../../bin/ls ~/bin
$ ls -l ~/bin/ls
lrwxrwxrwx   1 nicku  nicku  15 Dec 16 22:11 /home/nicku/bin/ls -> ../../../bin/ls
```

The idea is that the *source* name you type in your `ln -s` command is what you want to see on the right side of that arrow `->` when you list the link with `ls -l`.