Discussion comments

Linux Packages

We discussed a few features of the commands available to analyse and describe packages that a distribution will provide to users to keep them up to date. A useful guide to start with is at https://serverfault.com/questions/96964/list-of-files-installed-from-apt-package

On Debian-based systems (eg. Ubuntu and its derivatives, such as Mint) the graphical user interface to the update system is through the Update Manager, or for more functions, Synaptic. These will use lower level commands that can be used directly on the command line, viz. `apt` and `dpkg`.

The command

```
dpkg --help
```

will tell you what options are available, very briefly. The man page at

```
man dpkg
```

has more information, but, I think, poorly presented.

Finding out about a package and installation

To know more about a package (eg. the package cinnamon) that has been installed on your system, then you can use

```
dpkg --status cinnamon
```

This will list details of the package's dependences and what sub-packages it is responsible for, as well as other things about it.

The command

```
dpkg --get-selections > mypackages
```

will list all installed packages and place the list in a text file `mypackages`.

Transferring to a new system

This can be used to transfer the list to another installation in order that they can be synchronised. If I understand the man page correctly (and it is not clear), then the following sequence of commands on the new machine should accomplish this, after you have transferred the text file `mypackages` over to the new system. (This has not been tested!)

```
sudo apt-cache dumpavail | sudo dpkg --merge-avail
sudo dpkg --clear-selections
sudo dpkg --set-selections < mypackages
```

```
sudo apt-get dselect-upgrade
```

Associating files with packages

To discover what files a package supplies, or which package supplies a given file, the you will need a different command, `apt-file`. This is not installed by default, so first you will need to install it, either through `synaptic`, or the command:

```
sudo apt-get install apt-file
```

Then you must update its information to match the state of your system and repositories:

```
sudo apt-file update
```

After that, you are in a position to list the files a package can supply. The package does not have to installed for it list the files. For example, for one that is installed, and one not:

```
apt-file list cinnamon
```

```
apt-file list apache2
```

And to discover which package can supply a file name, do something like this:

```
apt-file search /usr/share/iptables/iptables.xslt
```

This is particularly useful if you find that a program does not work and is complaining it is lacking some file (for example a font). This will tell you which package to install to get the file. You do not have to give the whole filename, part is good enough, although you might find a long list of partial matches, one of which is what you are looking for:

```
apt-file search BabelStone
```

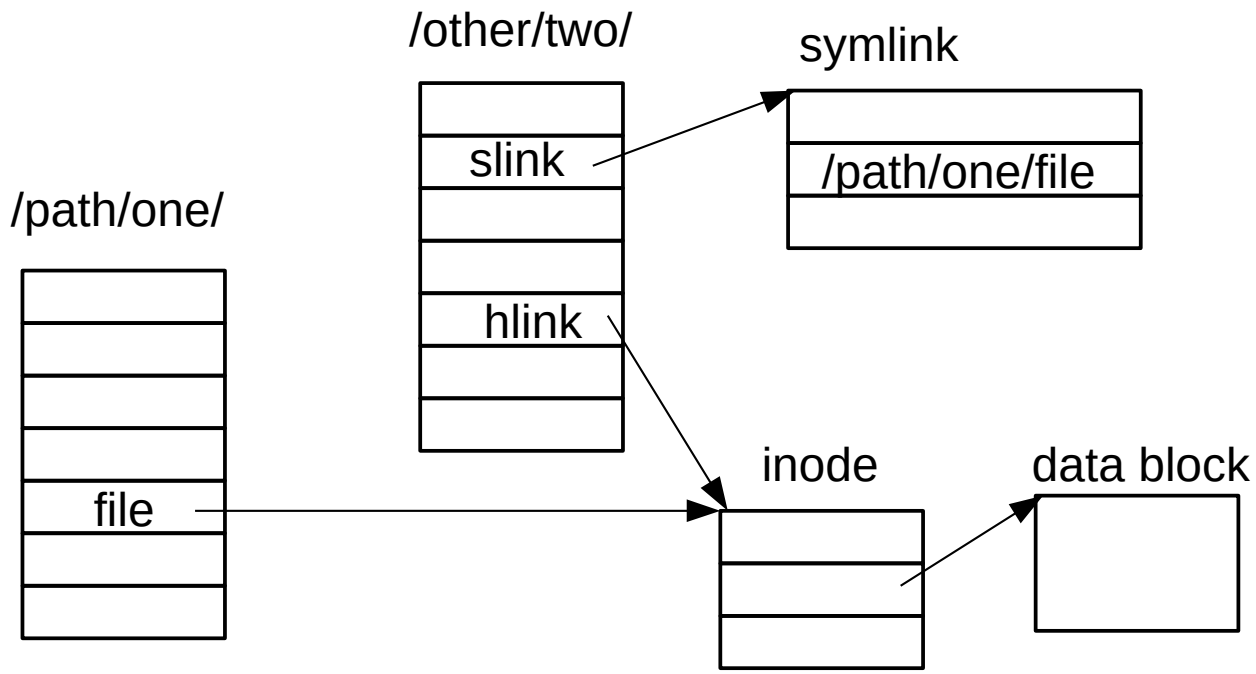
Unix Links (Symbolic and Hard)

Links are the Unix way of establishing synonyms for files, and so can be used to share files and folders between users, or between systems, etc.

Basic information can be found here: https://www.geeksforgeeks.org/soft-hard-links-unixlinux/

Links come in two forms, hard and soft. Soft links are also known as symlinks or symbolic links. I will describe both of these, but the one you will almost certainly need are symbolic links.

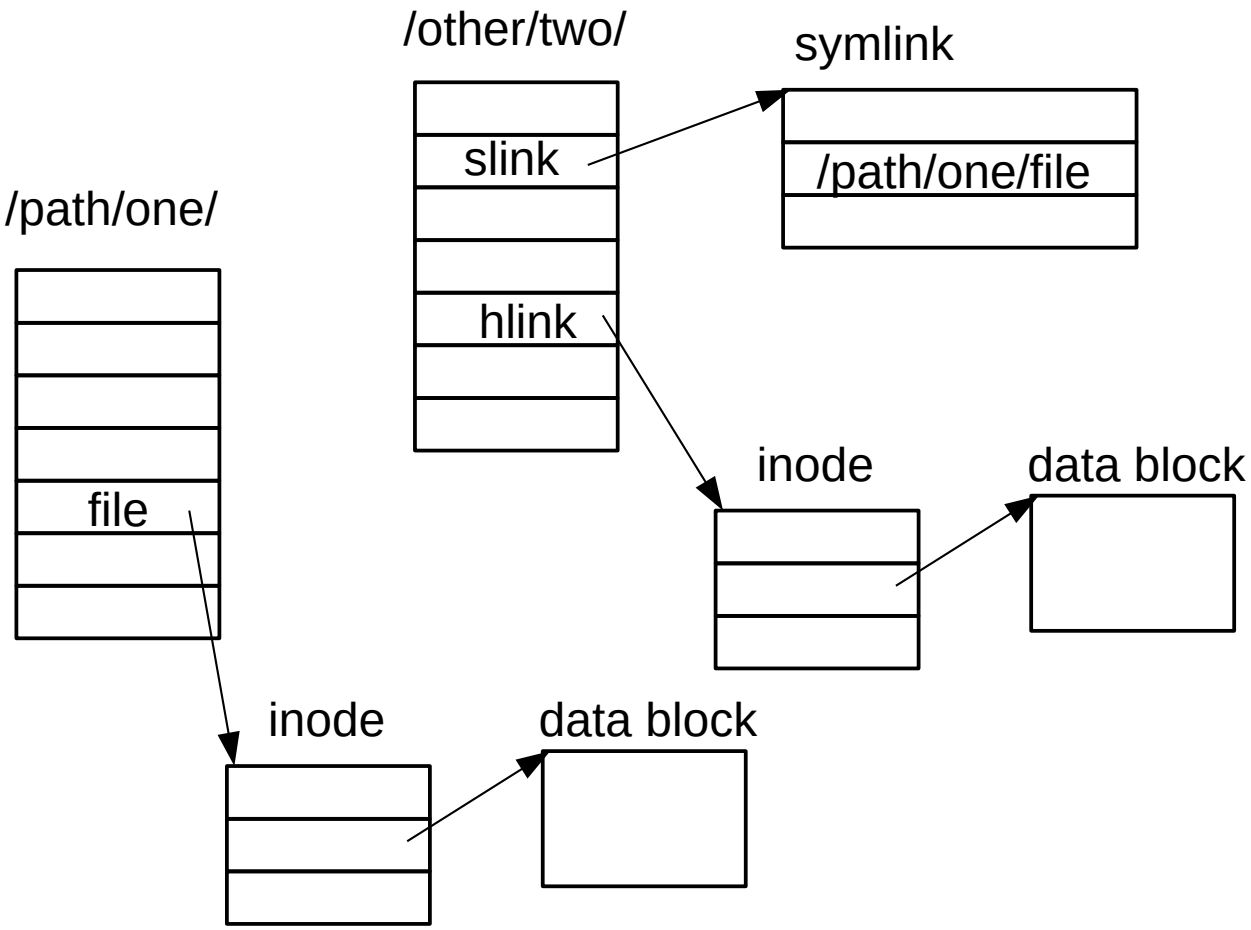
Consider the following situation. There is a folder (aka directory) at `/path/one/`, which contains file named ‘file’. The directory in reality will contain only a pointer to a data area called an inode, which contains information about the file such as its size, and the data area on disk where the data for the file really resides. This is represented in the diagram by a pointer to the data block. Although the physical realisation will differ from file system to file system, this abstraction is the same for all Unix-style file systems. It is not the same for other operating systems and their files.



There is another directory at `/other/two/` which contains links to the same file.

The hard link (`hlink`) is a direct reference to the very same inode and hence data block for the file. The soft, or symbolic, link (`slink`) however, refers to a descriptor containing the actual path to the file. At the moment, both links and the original file will see the same inode and hence the same data block and see the same contents of the file.

Now consider what happens when the file is updated via its real name. Almost all applications will create a new data area containing the new updated contents, and so we will get the following situation.



The file, when accessed by path one will now see the new data as expected. The symbolic link will also go to the name of the file and hence also see the updated file. However, the hard link will still be referring to the old data area and see the old contents. Hard links are the way most, if not all, snapshots are implemented in some backup systems. Because hard links need to interpret inodes, they can be used only within the same file system as the target file.

Typically, though, a normal user will always wish to see the current state of a file, and so a symbolic link is appropriate. Also, typically, they may well wish to refer to a file in another partition, and hence another file system, so a hard link would not then be possible.

To set up a symbolic link, most file managers supply like this: bring up the file or folder in one tab or window of the manager. Then bring up the directory, or folder, where you want the symbolic link to sit, in another tab or window. The hold down the CTRL and the SHIFT keys while dragging the target file or folder into the directory where you want the link to be.

Off-topic aside. The difference between references and contents was known to Lewis Carroll in *Through the Looking Glass* when the White Knight wants to sing a song to Alice:

> "… The name of the song is called '*Haddock's Eyes*'"

> “Oh, that’s the name of the song is it?” Alice said, trying to feel interested.
>
> “No, you don’t understand,” the Knight said, looking a little vexed. “That’s what the name is *called*. The name really *is ‘The Aged Aged Man’*”
>
> “Then I ought to have said ‘That’s what the *song* is called’?” Alice corrected herself.
>
> “No, you oughtn’t: that’s quite another thing! The *song* is called *‘Ways and Means’*, but that’s only what it’s *called*, you know!”
>
> “Well, what *is* the song, then?” said Alice, who was by this time completely bewildered.
>
> “I was coming to that,” the Knight said. “The song really is *‘A-sitting on a Gate’*: and the tune is my own invention.”

So now you have no excuse when you confuse names, addresses, references, and contents.

Making Forms with LibreOffice

The standard way of transferring a text file between users is to use a file in the format of a PDF. If you want the other party to fill in a form, then a PDF is still the best way to do it to ensure it is not dependent on the system they use.

In LibreOffice Writer, bring up the toolbar containing the various types of fields for forms by

```
View → Toolbars → Form Controls
```

You can now select the icon for the type of field you want, and drag it into the document where you want to place it, such as text fields, check boxes, numeric fields, etc. After placing it, you can resize by dragging the usual boundary markers.

Selecting the form field and right clicking, will bring up a further dialog with lots more options to tailor the form to what you actually want it to be, but I have not had time to investigate further.

You can save the file as a normal text file is the usual way.

To prepare a PDF to send elsewhere, then merely go to File → Export As → Export as PDF and follow the dialog. The resulting form can then be read, edited, and saved by any PDF reader.