Organisation
DIY package management in /usr/ local
STOW IT!

The */usr/ local* tree can easily become a tangle, but as Bruce Richardson explains this needn't be the case. GNU Stow is a simple application, which keeps it organised by allowing each piece of software to be installed into a separate directory tree

Up the creek without a package

Modern Linux distributions have sophisticated package management tools. Large and complex applications can be installed or removed with a single command or mouse click. Where you require non-standard options a few tweaks to the source package will usually give you what you want. Package management tools do your housekeeping for you and (when the packages are built to a well-planned policy) ensure that the various components of your system interoperate and function consistently.

Sometimes though, the app you want may not be available as a package, the source package may not be flexible enough or you may want to work with the latest cvs source. Whatever the reason may be, you find yourself working with the unpackaged source or binary files.

For most Linux users, this is not a daunting task. Typically, you unpack the tar-ball into */usr/ local/* src and run through some variation of the following:

```
# ./configure --some-option --some-other-option
# make
# make install
```

At the end of which the application should be safely installed into the */usr/ local* tree.

That is easy enough. At this point, however, you should ask yourself some questions:

- Where did all those files go? I can't use a package tool to get a simple list of what went where.
- Will it be easy to cleanly uninstall the application? Even if I have a list of all the installed files, what steps do I need to take to uninstall safely?
- Am I sure the application installed itself nicely and didn't break anything? This application wasn't packaged for my system and the developers may not have been as careful as they should have been.

These questions should worry you. The more applications you install like this, the harder it becomes to answer them. With unpackaged applications you have to do the housekeeping and maintenance yourself. This can turn into a nightmare.

```
/usr/local/ – The Land Where The Wild Things Are.
```

To quote the Filesystem Hierarchy Standard: "The */usr/ local* hierarchy is for use by the system administrator when installing software locally. It needs to be safe from being overwritten when the system software is updated". That is to say, it is the place to install software, which is not part of the standard system. In practice this means software that has not been pre-packaged using your distribution's packaging tools.

The */usr/ local* hierarchy is essentially a twin of the */usr* hierarchy, with bin, sbin, lib (and so on) subdirectories. A tar-balled application will usually install itself entirely within this hierarchy, unless you specify some other location. Precisely what goes where (docs to */usr/ local/ doc* or */usr/ local/ share/ doc*?) varies according to the developer's whim. If you are lucky, you may be able to place things exactly where you want them by passing the correct options to the configure script.

A simple solution

Stow offers a way to organise the */usr/ local* hierarchy, avoiding tangles and breakages. This is done by installing each application into its own corralled directory tree, and then creating symlinks to the application files. To install an application with Stow, follow this sequence:

- Create a destination directory for your new application. */usr/ local/ stow/* appname is traditional (and logical).

Further advantages

Having un-stowed an application, there is no need to delete it. You could, for example, keep two or more instances of an application in /usr/ local/ stow (different versions, perhaps, or compiled with different options) and switch between them at will. Just install them to different target directories.

The fact that Stow keeps applications in their own hierarchies makes them portable. You can quickly copy stowed apps between machines by making a tarball of the app's directory tree and un-tarring it into the Stow directory of the target machine.

Stow prevents applications from overwriting each other's files. Before it creates any symlinks it checks to see if any proposed links would overwrite existing files. If a conflict is found, Stow does not proceed.

- Install the software into this directory in such a way that files which would normally go into */usr/ local/ bin* are placed in */usr/ local/ stow/ appname/ bin*, files for */usr/ local/ share* go into *appname/ share* and so on. For tips on how to do this see the section called Installing to the target directory.
- Then simply do:

```
# cd /usr/local/stow
# stow appname
```

In the third step Stow moves recursively through the appname/ tree. For each file in *appname/ bin* a symlink is created in */usr/ local/ bin*, for files in appname/ doc links are created in */usr/ local/ doc* etc.

What was the point of that, you may ask? It's more laborious than the usual method and the installed application doesn't work any faster. The advantage becomes clear, however, when you come to uninstall the stowed app. Here's the entire procedure:

```
# cd /usr/local/stow
# stow -D appname
```

This removes all symlinks to the application. You are then free to delete the */usr/ local/ stow/* appname, knowing that you are deleting all the application's files and only those files.

Installing Stow

Stow is a Perl script and Perl is the only prerequisite. It should work with Perl 4 or Perl 5.

A Stow .deb package is available as part of the standard Debian distribution. Mandrake is the only rpm-based distribution for which we could find a Stow package. On any other set-up you will need to download the tar-ball from *ftp://ftp.gnu.org/gnu/stow*.

For the obsessive compulsives amongst you, it is possible to install both Perl and Stow as stowed applications:

- Install Perl into /usr/ local/ stow/ perl
- Install Stow into /usr/ local/ stow/ stow
- Now simply:

```
# cd /usr/local/stow
# perl/bin/perl stow/bin/stow perl stow
```

Installing to the target directory

Many source tarballs are designed to be relocatable. This means that you can change the base directory – the "prefix" – into which the application is installed, usually by passing a --prefix=desiredlocation option to the configure script (*/usr/ local* is usually the default). You might think that this is how you should install to the Stow target directory, but you would be wrong.

Things to watch

Stow attempts to create as few symlinks as possible. If it can link to a directory rather than the files within it, it will. So if the target directory contains a *lib/ data* directory but there is no data directory in */usr/ local/ lib*, Stow will create a symlink to the data directory, thus importing all its contents with only one link.

If you later use Stow to install another application which also includes a *lib/ data* directory, Stow will resolve the conflict by replacing the symlink with an actual */usr/ local/ lib/ data* directory and then populating that with symlinks to both applications.

Imagine what would happen, however, if you were to install the second application directly, not using Stow. The installation procedure would simply follow the symlink and install files directly into the first application's target directory. If you ever un-stow the first application, some files belonging to the second application would be unstowed with it.

To avoid this problem (a) make sure that */usr/ local* contains all the standard setup directories (bin, sbin, share and so on) and (b) use Stow to install all local applications, if possible.

Another gotcha is that ldconfig ignores symbolic links when scanning for libraries. If a stowed app includes libraries you may need to add some symlinks of your own in */usr/ local/ lib*.

Always un-stow an application before making any changes to the contents of the target directory. Re-stow after the changes are made. Otherwise you risk broken links.

- The app won't be run from its target directory. It will be run from its apparent location, as created by the symlinks.
- If the app uses shared resources, it will look for them in the prefix tree. If you set the prefix to be the Stow target directory, the app won't be able to find any shared resources because the target directory contains only files belonging to the application itself.

Instead you must let the application think it will be installed to */usr /local* (which is usually the default anyway) but divert the actual installation into the target directory. One way is to run

```
# make install prefix=/usr/local/stow/targetdir
```

rather than just

```
# make install
```

but this will not work for every application. See the Compile-time and install-time section of the Stow documentation for a detailed discussion of this issue.

Lastly...

My experience has been that while veteran Linux users tend to be familiar with Stow, newer users have usually not discovered it. This is a shame as it makes the potentially dangerous wilderness of */usr/ local* a much safer place. Properly used, Stow can help you keep your local hierarchy as tidy and well organised as the standard parts of your system.