Introduction

I've recently gotten tired of spamming my NeXTstation with software. Instead, I'd like to use its packaging system to create pacakges out of the things that I build. This has several advantages:

- I can easily keep track of everything that's been installed onto the system.
- I can easily remove and install software.
- If I get another NeXT (or have to reinstall on this one), it is much simpler to just copy over a bunch of packages and install than it is to rebuild everything.
- I can easily share with other NeXTstation owners.
- It gives me knowledge of another Unix.

After some searching around, I was unable to find any documentation on the system about how to make packages. However, I was able to find a csh script in /NextAdmin/Installer.App called 'package'. After a brief examination of that script, I figured out that it was used to make packages.

It turns out that making a NeXTSTEP package is pretty easy. In this article, I will cover the basics. Later on, I'll talk about the scripts that you can embed in your package, multiple architectures, and maybe even multiple languages. So let's get on to a description of the NeXT package format.

Package Format

A NeXT package is simply a directory that fits the name foo.pkg. When you download a package, it can come archived in any way that the packager desires; the most common are tar-gzipped and tar-compressed. The contents of that directory make up the package. A simple package will have the following files in it:

- foo.bom - The 'bill of materials' file. Among other things, this file contains a listing of the contents of the package. You'll need this if you wver want to remove the package.
- foo.info - Information for Installer.App (the NeXTSTEP pacakge manager). This tells Installer.App what the package is called, what version it is, a description, the default installation location, and more. When you go to install a package, all the information that you see is given by this file.
- foo.sizes - Information for Installer.App manager). This tells Installer.App how many files are present, the compressed size of the package, and the installed size of the package.
- foo.tar.Z - The tarred up and compressed contents of the package; This is the files and programs that get installed onto the system.

Building a Package

Now that you've got a basic understanding of what goes into a package, let's create one. The first thing to do is figure out what you want to package. For this example, I'll be using 'diff', which takes files and finds the differences between them. diff is a common Unix utility used by software developers.

Building the Software

First, I download the diff source, extract it, and build it. The most important aspect of this is choosing an installation prefix. I prefer /usr/local for anything that I build. The distributor installs things with a prefix of /usr, but software added by the local administrator should most likely go into /usr/local. For big NeXTSTEP graphical things, I would probably choose /LocalApps.

The other details of building diff are unimportant for this example. Once it has been built, you should then install it to the system. 'make install' will usually do it. Here comes one important thing to note about the NeXTSTEP packaging system. When you create a package, all you really do is create a fake root tree and copy into that tree everything that should go into the package. Since you specify an installation prefix for your package, you should leave that off.

Additionally, you should pay careful attention to the permissions and ownerships of the files that you copy into this package tree. Always copy things over as root and make sure that permissions are set properly. 'package' does nothing to modify the permissions and ownerships of the files in a package, so make sure that they are correct.

Creating the Package Root

Continuing with our example, doing a make install on the diff source will install the following:

- /usr/local/bin/diff
- /usr/local/bin/diff3
- /usr/local/bin/sdiff
- /usr/local/bin/cmp
- /usr/local/man/man1/cmp.1
- /usr/local/man/man1/diff.1
- /usr/local/man/man1/diff3.1
- /usr/local/man/man1/sdiff.1
- /usr/local/info/diff.info
- /usr/local/info/diff.info-1
- /usr/local/info/diff.info-2
- /usr/local/info/diff.info-3
- /usr/local/info/diff.info-4

I would then make a directory called /tmp/root to serve as the package building tree. Think of /tmp/root as taking the place of /usr/local for all the files listed above. Therefore, I would want to make the following directories: /tmp/root/bin, /tmp/root/man/man1, and /tmp/root/info. Then, just copy all the files from the running system into your package tree. After you're done, you would then have a tree that looked like this:

- /tmp/root/bin/diff
- /tmp/root/bin/diff3
- /tmp/root/bin/sdiff
- /tmp/root/bin/cmp
- /tmp/root/man/man1/cmp.1
- /tmp/root/man/man1/diff.1
- /tmp/root/man/man1/diff3.1
- /tmp/root/man/man1/sdiff.1
- /tmp/root/info/diff.info
- /tmp/root/info/diff.info-1
- /tmp/root/info/diff.info-2
- /tmp/root/info/diff.info-3
- /tmp/root/info/diff.info-4

Writing the diff.info File

Double check that all the permissions and ownerships are right. And you're basically done with the hard part. Now you'll need to write the info file; in this case, that would be named diff.info. Where you put this file is not important, but I prefer to just throw it into /tmp. The info file contains a keyword, followed by whitespace, followed by a value. These are the most common elements of an info file:

- Title - The name of the package. This appears in Installer.App when you go to install the package.
- Version - The version of the software in the pacakge. This also apepars in Installer.App when you install the package.
- Description - A description of the software package. This also appears in Installer.App when you install the pacakge.
- DefaultLocation - The default prefix for the package to go. I would set this to /usr/local for all but the biggest programs.
- Diskname - What to call the package directory.
- Relocatable - Is the person installing allowed to override the DefaultLocation with a different location? Values are YES or NO.
- Application - Does the package contain an application? Values are YES or NO. I haven't figured this one out yet, so I just use YES.
- LibrarySubdirectory - I haven't figured this one out either, so I just use Standard. I don't know what values are valid, either.

Here's the diff.info file that I'd write up for the diff package. Note that I make the whitespace between keyword and value a tab, but I don't know if it has to be.

```
Title          diff utilities
Version        2.7
Description    The GNU diff utilities find differences between files (to make source code patches, for instance).

DefaultLocation      /usr/local
Diskname             diff
Relocatable          YES
Application          YES
LibrarySubdirectory  Standard
```

Running 'pacakge'

Having created the package tree in /tmp/root and having written the diff.info file, I'm now ready to create the package. 'package' has two required arguments: the package tree, and the location of the info file. Call package like so:

```
$ cd /tmp
$ /NextAdmin/Installer.App/package /tmp/root/ /tmp/diff.info
```

You'll see package output some messages as it works on the package:

```
Generating Installer package ./diff.pkg ...
        creating package archive ... done.
        copying diff.info ... done.
        generating bom file ... done.
        generating sizes file ... done.
   ... finished generating ./diff.pkg.
```

And package will drop the diff.pkg directory into the directory you called it from. That's why I changed into /tmp first. If you examine the resulting diff.pkg/diff.tar.Z, you'll see that package strips the /tmp/root from the beginning of every file. When you go to install the package, the DefaultLocation (or location that the installer specifies, if Relocatable) will get tacked to the beginning of each file.

Distribution

You may now install the package onto your system if you'd like to keep track of it. Then, simply archive the package using your favorite archiver (I prefer tar-gzipped) and you're ready to distribute it. Here's how I would package up that diff.pkg directory:

```
$ cd /tmp
$ tar -cvf - diff.pkg | gzip -9c > diff-2.7-N.tar.gz
```

I've chosen that name because it shows what's in the archive, what version the software is, and for what architecture it was compiled (N - NeXT hardware).

Cleanup

After archiving the package, you can delete the info file, the root directory, the diff.pkg directory, and any source that you have laying around. Just don't delete the archived package, or you'll have to do all this stuff over again.

Closing

That's all for this introduction to the NeXTSTEP packaging system. Later, I will be talking about installing and removing packages, having control scripts in your packages, and an introduction to multiple languages and architectures in the same package. I hope you've enjoyed this article.

If you have any comments or corrections for this article, or would like to fill in the gaps in my knowledge, feel free to send me an email. I can be reached at: chris@bangmoney.org.