The install target should call the install program, which creates directories or copies a files with specified file permissions. For example, a binary can be copied into the destination directory by adding

```
install -d -m 755      $(DESTDIR)/usr/bin
install -m 755 amazing $(DESTDIR)/usr/bin
```

to the Makefile. In this example, the first command creates `/usr/bin` if it does not exist and the second copies the binary file with the appropriate file permissions for an executable program.

Creating .deb and .dsc files

The outputs from the packaging process are files of two types:

• `.deb` - binary builds of a program that can be directly installed on Raspbian.
• `.dsc` - source code of a program, ready to be built

We'll create these files using the debuild program. In common with several other packaging commands, debuild uses various helper programs to do its work. One of the disadvantages of this arrangement is that documentation is often spread across several locations. For example, the manual for debuild doesn't describe all of the options that it is parameterised by — some of them are only described by the helper program documentation. A good place to start looking for their meaning is in the documentation for dpkg-buildpackage (which in turn calls a whole team of other helpers).

The debuild commands that we need are these:

```
debuild -S -Ipackage
debuild -Ipackage
```

The first command builds only a .dsc source package (using the -S option). The second builds both a .dsc and a .deb. In both cases we ignore the package directory (via the -Ipackage option) as this is where we'll put the files generated by debuild.

We use the source-only build when we're targetting an Ubuntu PPA (see above) — these only accept source packages, building a .deb of their own from the source.

Examples
SimBaMon and BlinkIP

This section gives package file and Makefile code examples for the script programs described in our pages on the SimBaMon simple battery monitor and the Blink my IP daemons.

```
mopi prototype 5
```

Below are summaries of the debian files used to build these packages. All of the code is on GitHub, including the Makefile.

The control file (the basic details of the package, its version and dependencies):

```
Source: simbamond
Section: admin
Priority: optional
Maintainer: Hamish Cunningham
http://gate.ac.uk/hamish/  <hamish@gate.ac.uk>
Build-Depends: debhelper (>= 8.0.0),
devscripts,
gawk
Standards-Version: 3.9.3
```
Homepage:
http://pi.gate.ac.uk/pages/mopi.html#simbamon
Vcs-Git: git://github.com/hamishcunningham/pi-tronics.git
Vcs-Browser:
https://github.com/hamishcunningham/pi-tronics/tree/master/simbamon

Package: simbamond
Architecture: all
Depends: bc, ${misc:Depends}
Recommends: gpio
Description: SimBaMon, a Simple Battery Monitor
SimBaMon is an open source Linux daemon for monitoring battery levels;

See
http://pi.gate.ac.uk/pages/mopi.html#simbamon

The changelog file (the changes present in each version):