RPM vs DEB

Seth Kenlon

Since most of the software used on a GNU Linux system is free to use, share, and modify, it was common when Linux was first being developed to install software straight from raw source code. As Linux became more popular with people not interested in the finer points of building applications with Makefiles and C Compilers, a packaging system was created by Red Hat Linux, known as RPM ("RPM Package Manager"). It provided a way for users to quickly install applications with pre-built packages from distributors like Red Hat.

The developers of the Debian distribution also developed a packaging system, called APT ("Advanced Packaging Tool"), that would not only install an application but would also install all of the support libraries and smaller programs that the application replies on in order to run. Since APT was developed, the same improvements have been incorporated into RPM (actually, into its user-friendly frontend, yum ("Yellow Dog Updater, Modified").

Even though both methods of installing applications have the same purpose, they are both in wide use. Since the tools are intricately woven into the way a Linux distribution is built and functions, a user generally uses whatever package manager happens to come along with the Linux system they install.

RPM

In addition to Red Hat and many others, Fedora and openSUSE all use RPM for application installation. Getting an RPM is typically done in two different ways; from the distribution's online RPM servers ("repositories") or from third-party sites.

The first option is the most secure, since Linux distributions typically check

Figure 1. *Adding new applications to a Linux desktop is easy*

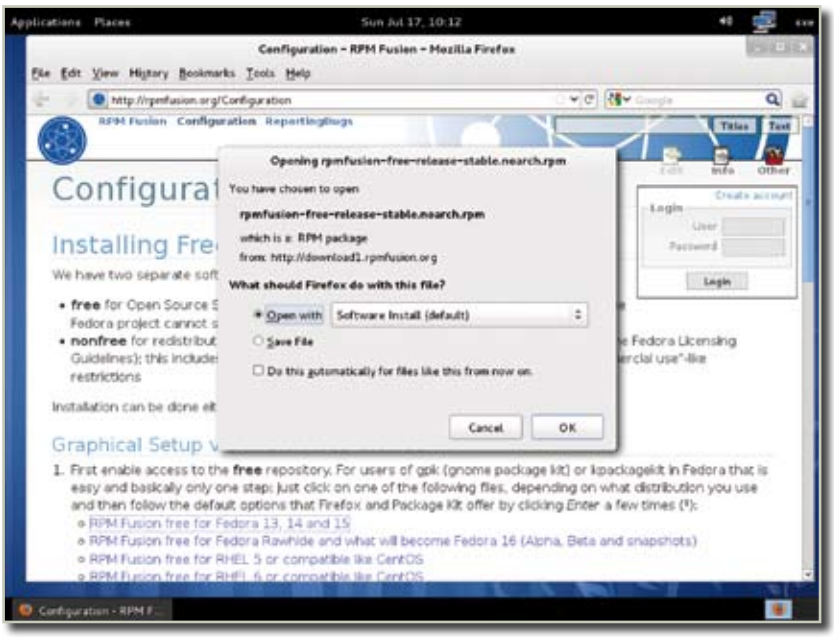

Figure 2. *Adding the rpmfusion.org repository for even more apps on Fedora 15*

their RPM files carefully. It is also the easier option to support, since the major Linux distributions are well-known for updating applications on a regular basis, whereas RPMs installed from random third parties don't necessarily come with any guarantee of a reasonable, much less timely, upgrade path.

The easiest way to access a distibution's RPM repository is through either Package Kit in Fedora or YaST in openSUSE. These are both, essentially, app stores from which you can pick and choose appealing applications to install on your Linux computer. Each RPM contains a "spec" file, detailing what version of the application it will install and also what other smaller applications will need to be installed in order for it to function. Of course, you won't have to worry about any of these details; Package Kit or YaST will resolve the dependencies by installing the necessary programs, and then the application itself.

Figure 3. *When all else fails, you can build your own installer packages*

If your distribution does not offer an RPM for an application that you need, you can still check trusted repositories, such as *rpmfusion.org* or *ccrma.stanford.edu/planetccrma/software/* for Fedora, or *packman.links2linux.org* for openSUSE. These repositories have a history with the community and are well-respected, trusted, and reliable. Many people would never think of running their distribution without a few extra unofficial repositories, and few would feel safe going out onto the Internet to unknown sites and allowing those sites to install software on their computers.

Building an RPM

RPMs, having been around for so long and being the choice of one of the most popular Linux distributions, are ubiquitous online. Some software projects, when packaging their software, do not provide an RPM but do include a .spec file in the source code. With this .spec file, you can actually construct your own personal RPM with only a few commands in a terminal.

First, of course, you'd need to download the source code (that is, the actual raw code that makes up the application, and not an installer file). These usually come in the for of a .tar.gz or .tar.bz2 archive, or sometimes a .zip archive.

Then you should set your system up for building RPM packages by installing the necessary tools:

```
$ su
<enter your password>
#    yum    groupinstall    ↵
   "Development Tools"
# yum install rpmdevtools
```

This installs the necessary software to compile the code you've downloaded into a running application, and the tools necessary to construct an RPM installer file. Now you can create a clean environment in which to build the RPM:

```
$ cd ~ && $ rpmdev-setuptree
```

This will create a new folder in your user directory, called rpm-

build, with folders inside of that called BUILD, BUILDROOT, RPMS, SOURCES, SPECS, and SRPMS. Happily, you don't actually need to know what any of that means in order to use it.

Next, you must unzip or untar the source code directory; keep in mind that not all source code will include a .spec file, but we'll assume for this article that the software (let's call it "foo", like the Unix gurus do) in our example does.

```
$ tar -xf foo-0.1.tar.gz && cd ↵
   foo-0.1
$  cp  foobar-0.1.spec  ~/  ↵
   rpmbuild/SPECS
$ mv ../foobar-0.1.tar.bz2 ~/ ↵
   rpmbuild/SOURCES
```

This places all the code in the proper location, so now all you need to do is run the RPM build commands:

```
$ cd ./rpmbuild/SPECS
$ rpmbuild -ba foobar- 0.1.spec
```

You'll get to watch some code scroll by your screen as the raw data is converted into an executable application. Once that's finished, you're free to install the RPM onto your system.

```
$ cd ../RPMS/$(uname -m)
$ su
<enter your password>
# yum localinstall foobar-0 ↵
   .1.$(uname -m).rpm
```

DEB

The .deb package format is used by most if not all Debian derivative distributions, usually with the installation itself occurring through APT. The DEB format is dependent upon a "control file" which is much like the .spec file of an RPM package, as it defines what software is required in order for your target application to run, provides a description of what you are about to install, and so on.

Installing a .deb is the same, basically, as an RPM file. Launch your package manager, which may be the Synaptic frontend if you are using Debian, or the Ubuntu One Software Store on Ubuntu. Find an application that you want to install, and click the install button. This automatically downloads the .deb file, processes it in order to install dependent support libraries, and then installs the application itself. You'll find the launcher for that application in the *Application* menu in GNOME, the Mint menu on Linux Mint, the *Application* screen on Ubuntu.

Figure 4. *Using the Ubuntu One software store to install .deb packages*

If you can't find an application that suits your needs in the distribution's repositories, then of course there are trusted third-party repositories just as on Fedora and openSUSE. Ubuntu has its "multiverse" repositories as well as PPA repositories, which are run and managed by individuals in the community. Debian has a number of repositories, one of the most common being *debian-multimedia.org* for multime-

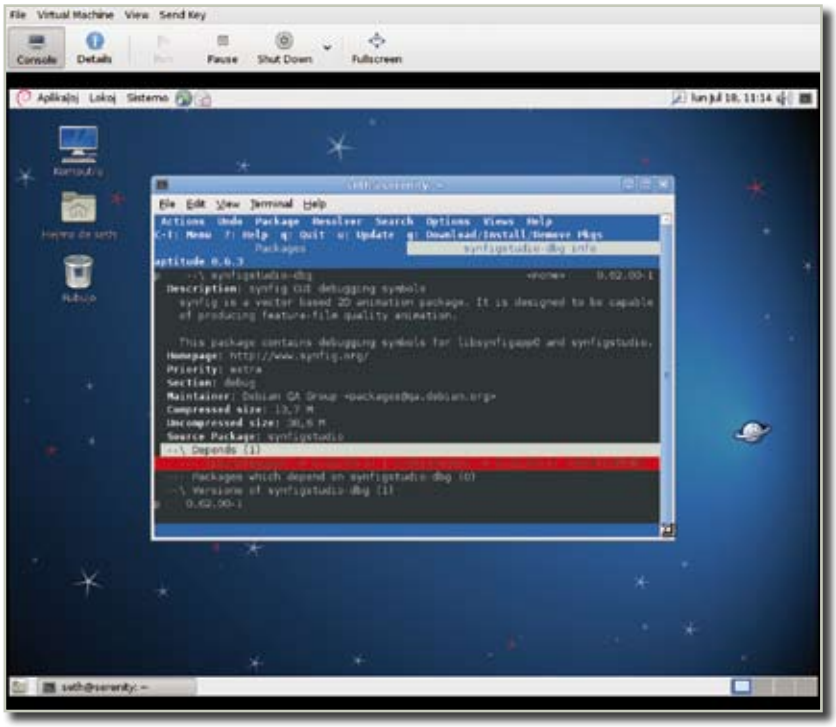

Figure 5. *Using Debian aptitude to install .deb files*

dia codecs and *kg-kde.alioth.debian.org* for KDE's desktop environment.

Downloading RPM and DEB Files

Downloading a .deb or .rpm file outside of trusted repositories is, of course, not necessarily recommended, but there are times when it is something you will want to do. For instance, you might find that your favorite application has just released a new version with exciting new features that you need to have. Rather than waiting for your repository to get that version of the application, you might go to the application's site and download the .deb or .rpm that they themselves have released.

The end result is that you have a .deb or .rpm file sitting on your desktop or in your *Downloads* folder; how do you install it?

Happily, with either an .rpm or .deb file, you can simply double-click on the file and the operating system will offer to install it via your usual package manager. If this does not happen, then you can also do it from the command line.

For an .rpm file:

```
su -c 'rpm -i ./foo-0.1.rpm'
```

And for a .deb file, you can use this command:

```
sudo dpkg -i ./foo-0.1.rpm
```

Converting RPM to DEB and DEB to RPM

In the event that an application you feel you must have is available in only one format and that one format is not the one you need, then you can try to convert between the two package formats with a command line tool called "alien".

Alien is a very simple tool to use, and the process is the same as installing a package from the command line, with one extra step.

First, download the package you wish to install. Now you have the package on your desktop or in your Downloads folder, so you need to convert it to a format that your distribution will understand.

If it is currently a .deb file and you wish to have an .rpm:

```
alien -r foo-0.1.deb
```

If it is currently a .rpm file and you wish to have a .deb:

```
alien -d foo-0.1.rpm
```

And then install as normal; for an .rpm:

```
su -c 'rpm -i ./foo-0.1.rpm'
```

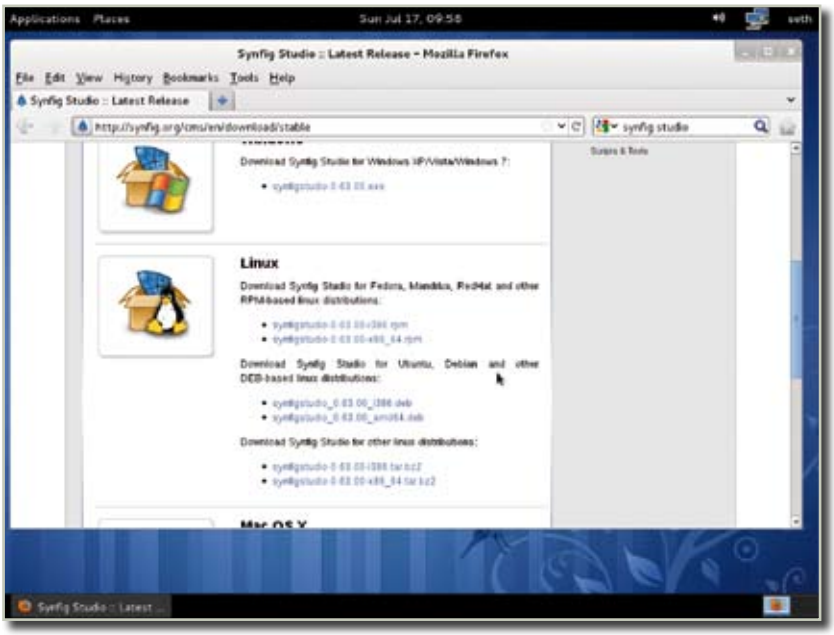

Figure 6. *Many applications have both RPM and DEB packages available from their homepage*

And for a .deb:

```
sudo 'dpkg -i ./foo-0.1.deb'
```

The only caveat here are the dependencies. Sometimes, because the installer package was really intended for, say, Ubuntu, but you wish to install it on Fedora, the conversion will work but the package's effort to ensure that your system has all the required software will fail. To avoid this, you can look at what the system requirements are and manually install the dependencies from your repository, and then install the converted package.

RPM vs DEB

While some people will tell you that rpm-based distributions are difficult to use, or that deb-based distributions are too confusing, and so on, the truth of the matter is that there really is no difference. The two formats do the same thing: ensure that an application that you wish to install will run on your computer, and then install the application.

The applications that you use to install the packages may differ a little; yum checks for updates unless you tell it not to, while aptitude requires a manual update of its cache, and so on, but in the end it all works toward the same goal.

There are other methods of installing software; some prefer compiling from source code, others use scripting tools like SlackBuilds, and still others use new paradigms altogether. The important thing is that you find applications from trusted sources, and use common sense when installing from unknown sites.

Properly managed, your Linux computer will remain a secure, finely-tuned operating system, with the ability to install and uninstall with far greater efficiency and precision than you previous operating systems ever did, so enjoy your rpm, or your deb, or raw code. ■