Package Management*

Hannes Hirzel

Based on *Package Management*† by

Algis Rudys

This work is produced by The Connexions Project and licensed under the
Creative Commons Attribution License ‡

Abstract

This module provides an overview how software packages are managed in Linux distributions. It covers the RPM and Debian formats as well as the Apt and yum tool. As an alternative it also discusses how to compile an application from the source files.

Software packages
A software package is a piece of software stored in an archive format together with catalog information (**metadata**) about the content. The metadata includes the name, the author(s), a short description, the version number and a list of other packages the package depends on. The package contains a group of files which are used by the same program, library or subsystem.
Package management system
A package management system is a system which allows packages to be installed, removed, and configured as a unit. Linux distributions are normally segmented into packages. With a package management system an installation may be tailored to specific needs.
Packages in Linux
There are several package types in use in Linux systems. A package type refers to the way the package is structured and how the packages are organized. Two common ones are **RPM** and the **Debian** package type. RPM is a package type created by the company RedHat. Normally a Linux distribution is based on only one package type. So there are RPM-based distributions like RedHat, Fedora, CentOS, Suse, Mandriva and Debian based distributions like Debian itself, Knoppix and Ubuntu. RPM packages have the suffix *.rpm whereas Debian packages have the suffix *.deb. `apt` is a collection of command line tools for managing Debian packages, wheras `yum` is used to deal with RPM packages.

1 RPM and yum

RPM refers to

- a software package format
- software packaged in this format

*Version 1.6: Jun 18, 2012 8:26 am -0500
†http://cnx.org/content/m10468/2.3/
‡http://creativecommons.org/licenses/by/3.0/

- a tool called 'rpm' to deal with these packages

RPM was developed by the company Red Hat, it is now also used by Suse, Mandrake and other distributions. In addition to supporting package installation and removal, it provides support for resolving dependencies and conflicts with other packages.

There is a command line tool `rpm` which may be used to install packages.

```
rpm -ihv file-xxx.xx.xx-i386.rpm
```

or

```
rpm -Uhv file-xxx.xx.xx-i386.rpm
```

where 'h' is a progress indicator, 'v' means 'verbose', 'i' is 'install' and 'U' is 'install and upgrade'. So it is advisable to use 'U' alway instead of 'i' disregarding whether you are going to install or upgrade, unless you are installing a new kernel. So, again, use this: `rpm -Uhv file-xxx.xx.xx-i386.rpm`

Multiple packages

You may install several packages at once. Just enter them in one line separated by a space. Or you can use something like this: `rpm -Uhv *.rpm` to install the packages contained in all the rpm files in the working directory.

Removing a package

To remove a package use this: `rpm -e name_of_package` Note the difference - to install a package you enter the file name, to remove it - the name of the package.

yum is easier to use

There is an easier way to deal with rpm packages. Use the `yum` tool.

```
yum install name_of_package
```

The 'yum' command uses 'rpm' but keeps track of all dependencies. For instance, if you want to install the vlc player, use:

```
yum install vlc
```

To remove a package with all dependencies use this

```
yum remove vlc
```

All the commands given above have to be done as **root user**.

2 Dpkg and apt

Dpkg - Debian package manager

dpkg is the software at the base of the Debian package management system. dpkg is used to install, remove, and provide information about `.deb` packages [Wikipedia[1]]. Like RPM, it has support for dependency and conflict resolution. It also supports weaker dependencies (that is, a package can "suggest" or "recommend" another).

dpkg is a low level tool. Most users do not use it directly. They use `apt-get` instead or a graphical front end like **synaptic** or the Ubuntu software center.

Apt

Apt (Advanced Packaging Tool) is a front end for dpkg to make it easier to use dpkg. To install a package use the command

```
apt-get install packagename
```

[1] http://en.wikipedia.org/wiki/Dpkg

It downloads, installs, and configures the package and if necessary all other packages on which the package to be installed depends. Often you have to install packages in superuser mode

```
sudo apt-get install packagename
```

The 'apt' is similar in function to 'yum' but it is more powerful.
Difference between package file name and package name
dpkg deals with the package file name which might include the version number and the platform the package is for, e.g. i386. The package name to the contrary is short. It is independent. For this reason apt is easier to use.
Example: install the Apache web server.
On a command-line type `sudo apt-get install apache2`. Then answer the questions.
Example: install Abiword

```
sudo apt-get install abiword
```

and to remove it

```
sudo apt-get remove abiword
```

In addition Apt may be used to query or search the package database by using the `apt-cache` tool.

apt-cdrom is used to add a new CDROM or DVD to the list of available packages and sources. With apt-cdrom you may specify a different folder than a CDROM, using the -d option (i.e. a hard disk or a USB pen drive).

Apt may be used for upgrading and installation. By running the command `apt-get upgrade` the latest version of all out-of-date packages will be downloaded and installed. `apt-get dist-upgrade` goes one further, allowing one to upgrade entire releases.

3 Compiling Packages by Hand

If there is no package available in the distribution, for legal reasons (license) or because of other reasons people as well compile the software from the source files on Linux machines. For example there are some packages which are illegal to distribute in binary form, but may be distributed legally in source code form.

Source packages tend to be released as `.tar.gz` files. These are compressed tar archives. They can be uncompressed and expanded with the command `tar -xzvf file.tar.gz`. Most such packages have a README or INSTALL file at the top level which describes how to compile and install the package.

```
tar zxvf progname-x.x.tar.gz
```

Enter the newly created directory

```
cd progname-x.x
```

Then run the 'configure' script

```
./configure
```

Then compile the program

```
make
```

Then install it

```
make install
```

In the absence of errors, this will result in the application being installed as desired in the package's default location (generally under `/usr/local`). This and other compile settings can generally be adjusted by passing options to the configure script (the `-help` option will list all available options). The installation command has to be done as root user.

3.1 References

The following links indicate the type of package management the various Linux distributions use.

- List_of_Linux_distributions[2] (Wikipedia)
- Distrowatch[3]

3.2 Acknowledgments

Thanks to Alexander Vinokurov for providing the rpm examples and for general feedback.

[2] http://en.wikipedia.org/wiki/List_of_Linux_distributions
[3] http://distrowatch.com/