Learn Linux, 101: Debian package management

Add new software and keep your system current

Ian Shields
Senior Programmer
IBM

11 May 2010

Learn how to install, upgrade, and manage packages on your Linux® system. This article focuses on the Advanced Packaging Tool, or APT, which is the package management system used by Debian and distributions derived from Debian, such as Ubuntu. You can use the material in this article to study for the LPI 101 exam for Linux system administrator certification, or just to explore the best ways to add new software and keep your system current.

View more content in this series

About this series

This series of articles helps you learn Linux system administration tasks. You can also use the material in these articles to prepare for Linux Professional Institute Certification level 1 (LPIC-1) exams.

See our developerWorks roadmap for LPIC-1 for a description of and link to each article in this series. The roadmap is in progress and reflects the latest (April 2009) objectives for the LPIC-1 exams: as we complete articles, we add them to the roadmap. In the meantime, though, you can find earlier versions of similar material, supporting previous LPIC-1 objectives prior to April 2009, in our LPI certification exam prep tutorials.

Overview

In this article, learn to use the Debian package management tools to manage the packages on your Linux system. Learn to:

- Install, reinstall, upgrade, and remove Debian binary packages
- Find packages containing specific files or libraries, even if the package is not installed
- Obtain package information like version, content, dependencies, package integrity, and installation status, even if the package is not installed

This article helps you prepare for Objective 102.4 in Topic 102 of the Linux Professional Institute's Junior Level Administration (LPIC-1) exam 101. The objective has a weight of 3.

Prerequisites

To get the most from the articles in this series, you should have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this article. Sometimes different versions of a program will format output differently, so your results may not always look exactly like the listings and figures shown here. In particular, much of the output we show is highly dependent on the packages that are already installed on our systems. Your own output may be quite different, although you should be able to recognize the important commonalities.

Introducing package management

Connect with Ian

Ian is one of our most popular and prolific authors. Browse all of Ian's articles on developerWorks. Check out Ian's profile and connect with him, other authors, and fellow readers in My developerWorks.

In the past, many Linux programs were distributed as source code, which a user would build into the required program or set of programs, along with the required man pages, configuration files, and so on. Nowadays, most Linux distributors use prebuilt programs or sets of programs called *packages*, which ship ready for installation on that distribution. In this article, you will learn about *package management* tools that help you install, update, and remove packages. This article focuses on *Advanced Packaging Tool*, or *APT*, the package management system used by Debian and distributions derived from Debian, such as Ubuntu. Another article in this series, "Learn Linux 101: RPM and YUM package management," covers the Red Hat package management tools.

Develop skills on this topic

This content is part of a progressive knowledge path for advancing your skills. See Basics of Linux system administration: Setting up your system and software

From a user perspective, the basic package management function is provided by commands. As Linux developers have striven to make Linux easier to use, the basic tools have been supplemented by other tools, including GUI tools, which hide some of the complexities of the basic tools from the end user. In this article and in the article on RPM and YUM package management, we focus on the basic tools, although we mention some of the other tools so you can pursue them further.

APT, RPM, and YUM (the latter two are package management tools for Red Hat systems) have many similarities. All can install and remove packages. Information about installed packages is kept in a database. All have basic command-line functionality, while additional tools can provide more user-friendly interfaces. All can retrieve packages from the Internet.

When you install a Linux system, you typically install a large selection of packages. The set may be customized to the intended use of the system, such as a server, desktop, or developer workstation. And at some time, you will probably need to install new packages for added functionality, update the packages you have, or even remove packages that you no longer need or that have been made obsolete by newer packages. Let's look at how you do these tasks, and at some of the related challenges such as finding which package might contain a particular command.

Installing Debian packages

Suppose you want to learn Lisp, and a colleague tells you to use the `gcl` command. You might try `gcl --help`, or you might try `which gcl`, or `type gcl`. But if your system can't find `gcl`, you might see output similar to that shown in Listing 1.

Listing 1. Missing gcl command

```
ian@pinguino:~$ gcl --help
-bash: gcl: command not found

ian@pinguino:~$ gcl --help
The program 'gcl' is currently not installed.  You can install it by typing:
sudo apt-get install gcl
gcl: command not found

ian@pinguino:~$ which gcl

ian@pinguino:~$ type gcl
-bash: type: gcl: not found
```

If you did not get the helpful suggestion from the second form of output in Listing 1, you might check back with your colleague to find out which package to install. Otherwise, you might just guess that the `gcl` command is in the gcl package. This is often a good guess, but not always the right one. We'll see later how to find the right package. In this case, you need the gcl package, and you install it using the `apt-get` command with the `install` option as shown in Listing 2. Note that `apt-get` will determine which extra packages you need to satisfy dependencies and will then give you a list of all the packages that will be installed. At that point, you are prompted to continue. In our example, we respond `y` to install gcl and the additional required package, libreadline5.

Listing 2. Installing gcl using apt-get

```
ian@pinguino:~$ sudo apt-get install gcl
[sudo] password for ian:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-2.6.31-14 linux-headers-2.6.31-14-generic
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  libreadline5
Suggested packages:
  gcl-doc
The following NEW packages will be installed:
  gcl libreadline5
0 upgraded, 2 newly installed, 0 to remove and 30 not upgraded.
Need to get 47.1MB of archives.
After this operation, 157MB of additional disk space will be used.
Do you want to continue [Y/n]? y
Get:1 http://us.archive.ubuntu.com karmic/main libreadline5 5.2-6 [140kB]
Get:2 http://us.archive.ubuntu.com karmic/universe gcl 2.6.7-45ubuntu1 [47.0MB]
Fetched 47.1MB in 1min 33s (502kB/s)
Preconfiguring packages ...
Selecting previously deselected package libreadline5.
(Reading database ... 142156 files and directories currently installed.)
Unpacking libreadline5 (from .../libreadline5_5.2-6_i386.deb) ...
Selecting previously deselected package gcl.
Unpacking gcl (from .../gcl_2.6.7-45ubuntu1_i386.deb) ...
Processing triggers for man-db ...
Setting up libreadline5 (5.2-6) ...
Setting up gcl (2.6.7-45ubuntu1) ...
install/gcl: Handling install for emacsen flavor emacs22
Loading 00debian-vars...
No /etc/mailname. Reverting to default...
Loading /etc/emacs/site-start.d/50dictionaries-common.el (source)...
Loading debian-ispell...
Loading /var/cache/dictionaries-common/emacsen-ispell-default.el (source)...
Loading /var/cache/dictionaries-common/emacsen-ispell-dicts.el (source)...
Loading /etc/emacs/site-start.d/50gcl.el (source)...
Wrote /usr/share/emacs22/site-lisp/gcl/add-default.elc
Wrote /usr/share/emacs22/site-lisp/gcl/ansi-doc.elc
Wrote /usr/share/emacs22/site-lisp/gcl/dbl.elc
Wrote /usr/share/emacs22/site-lisp/gcl/doc-to-texi.elc
Wrote /usr/share/emacs22/site-lisp/gcl/gcl.elc
Wrote /usr/share/emacs22/site-lisp/gcl/man1-to-texi.elc
Wrote /usr/share/emacs22/site-lisp/gcl/smart-complete.elc
Wrote /usr/share/emacs22/site-lisp/gcl/sshell.elc
install/gcl: Handling install for emacsen flavor emacs23
Loading 00debian-vars...
No /etc/mailname. Reverting to default...
Loading /etc/emacs/site-start.d/50dictionaries-common.el (source)...
Loading debian-ispell...
Loading /var/cache/dictionaries-common/emacsen-ispell-default.el (source)...
Loading /var/cache/dictionaries-common/emacsen-ispell-dicts.el (source)...
Loading /etc/emacs/site-start.d/50gcl.el (source)...
Wrote /usr/share/emacs23/site-lisp/gcl/add-default.elc
Wrote /usr/share/emacs23/site-lisp/gcl/ansi-doc.elc
Wrote /usr/share/emacs23/site-lisp/gcl/dbl.elc
Wrote /usr/share/emacs23/site-lisp/gcl/doc-to-texi.elc
Wrote /usr/share/emacs23/site-lisp/gcl/gcl.elc
Wrote /usr/share/emacs23/site-lisp/gcl/man1-to-texi.elc
Wrote /usr/share/emacs23/site-lisp/gcl/smart-complete.elc
Wrote /usr/share/emacs23/site-lisp/gcl/sshell.elc

Processing triggers for libc-bin ...
ldconfig deferred processing now taking place
```

From the output in Listing 2, you see that `apt-get` has read a package list from somewhere (more on that shortly), built a dependency tree, and determined that libreadline5 is a required prerequisite that is not currently installed. You will also notice the suggestion to install the separate package for the documentation, gcl-doc. After some additional summary information, including space usage, you are prompted to continue, and gcl is installed along with the prerequisite package. Debian packages usually have an extension of .deb, and you see that the packages are downloaded and unpacked as shown in the line:

```
Unpacking gcl (from .../gcl_2.6.7-45ubuntu1_i386.deb) ...
```

Suppose that, instead of installing a package, you simply want to find out whether the package depends on other packages. You can use the `-s` (for *simulate*) option on `apt-get`. There are several other options with equivalent function, such as `--just-print` and \`--dry-run`. Check the man pages for full details. Listing 3 shows what happens for a simulation of installing the gcl-doc package.

Listing 3. Simulated or dry-run install of gcl-doc

```
ian@pinguino:~$ sudo apt-get install -s gcl-doc
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-2.6.31-14 linux-headers-2.6.31-14-generic
Use 'apt-get autoremove' to remove them.
The following NEW packages will be installed:
  gcl-doc
0 upgraded, 1 newly installed, 0 to remove and 30 not upgraded.
Inst gcl-doc (2.6.7-45ubuntu1 Ubuntu:9.10/karmic)
Conf gcl-doc (2.6.7-45ubuntu1 Ubuntu:9.10/karmic)
```

Not surprisingly, the documentation does not have any prerequisite packages.

Package locations

In the previous section, you learned how to install a Debian package. But where do the packages come from? How does `apt-get` know where to download packages from? We mentioned that `apt-get` read a package list from somewhere. The starting point for that somewhere is /etc/apt/sources.list. The list tells `apt-get` where to look for packages, including from a CD-ROM, from your local file system, or over a network using http or ftp. You can add additional sources in the /etc/apt/sources.list.d directory.

Listing 4 shows the first few lines of /etc/apt/sources.list on my system. Note that the distribution CD on the first line is commented out (# in position 1). If you need to install a lot of new packages that may not have been heavily updated, it may be worthwhile to uncomment this and install from your distribution CD or DVD. If you have a broadband network connection or need a lot of updates, it may be more efficient to just download the additional packages at the latest level from the network sources that follow in /etc/apt/sources.list.

Listing 4. /etc/apt/sources.list

```
ian@pinguino:~$ cat /etc/apt/sources.list
#deb cdrom:[Ubuntu 9.10 _Karmic Koala_ - Release i386 (20091028.5)]/ karmic main restrict
ed
# See http://help.ubuntu.com/community/UpgradeNotes for how to upgrade to
# newer versions of the distribution.

deb http://us.archive.ubuntu.com/ubuntu/ karmic main restricted
deb-src http://us.archive.ubuntu.com/ubuntu/ karmic main restricted

## Major bug fix updates produced after the final release of the
## distribution.
deb http://us.archive.ubuntu.com/ubuntu/ karmic-updates main restricted
deb-src http://us.archive.ubuntu.com/ubuntu/ karmic-updates main restricted

## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team. Also, please note that software in universe WILL NOT receive any
## review or updates from the Ubuntu security team.
deb http://us.archive.ubuntu.com/ubuntu/ karmic universe
deb-src http://us.archive.ubuntu.com/ubuntu/ karmic universe
deb http://us.archive.ubuntu.com/ubuntu/ karmic-updates universe
deb-src http://us.archive.ubuntu.com/ubuntu/ karmic-updates universe
```

`Apt-get` and similar tools use a local database to determine what packages are installed. They can check installed levels against available levels. To do this, information on available levels is retrieved from the sources listed in /etc/apt/sources.list and stored on your local system. You use the command `apt-get update` to synchronize the information in your local database with the sources specified in /etc/apt/sources.list. You should do this before installing or updating any package, and always after modifying /etc/apt/sources.list or adding files to /etc/apt/sources.list.d.

Removing Debian packages

If you want to remove a package, you can use the `remove` option of `apt-get`. A simulated run is shown in Listing 5.

Listing 5. Simulated removal of gcl

```
ian@pinguino:~$ sudo apt-get remove -s gcl
[[sudo] password for ian:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-2.6.31-14 linux-headers-2.6.31-14-generic libreadline5
Use 'apt-get autoremove' to remove them.
The following packages will be REMOVED:
  gcl
0 upgraded, 0 newly installed, 1 to remove and 30 not upgraded.
Remv gcl [2.6.7-45ubuntu1]
```

Notice that the libreadline5 package that we installed as a prerequisite for gcl is not removed automatically, although one of the output lines tells us that this package, along with two Linux header packages, is no longer required. The `autoremove` function of `apt-get` (or the equivalent `remove` function and the `--auto-remove` option) will remove the requested packages, along with any packages that were installed as dependencies but are no longer required by any installed packages. This includes dependencies installed by packages other than the one or ones you are trying to remove, such as the linux-headers-2.6.31-14 and linux-headers-2.6.31-14-generic packages in our example. Listing 6 shows how to remove gcl, its dependency libreadline5, and the two unrelated Linux header packages that are no longer required.

Listing 6. Removing gcl and dependencies

```
ian@pinguino:~$ sudo apt-get autoremove gcl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  gcl libreadline5 linux-headers-2.6.31-14 linux-headers-2.6.31-14-generic
0 upgraded, 0 newly installed, 4 to remove and 30 not upgraded.
After this operation, 239MB disk space will be freed.
Do you want to continue [Y/n]? y
(Reading database ... 142327 files and directories currently installed.)
Removing gcl ...
remove/gcl: purging byte-compiled files for emacs22
remove/gcl: purging byte-compiled files for emacs23
Removing libreadline5 ...
Removing linux-headers-2.6.31-14-generic ...
Removing linux-headers-2.6.31-14 ...
Processing triggers for man-db ...
Processing triggers for libc-bin ...
ldconfig deferred processing now taking place
```

If you use the `autoremove` function of `apt-get` but do not specify any package name, then all unused packages that were installed as dependencies will be removed from your system. You can also use the `apt-get purge` option to remove configuration information. See the man page for more information.

Updating Debian packages

If you need to update an individual package, use `apt-get` with the `install` option again. Listing 7 shows how to update the already installed tzdata package on my system. Remember to run `apt-get update` before updating packages to make sure your local database reflects the latest available updates.

Listing 7. Updating a single package

```
ian@pinguino:~$ sudo apt-get install  tzdata
[sudo] password for ian:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be upgraded:
  tzdata
1 upgraded, 0 newly installed, 0 to remove and 29 not upgraded.
Need to get 679kB of archives.
After this operation, 0B of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com karmic-updates/main tzdata 2010i-0ubuntu0.9.10 [679kB]
Fetched 679kB in 1s (569kB/s)
Preconfiguring packages ...
(Reading database ... 124394 files and directories currently installed.)
Preparing to replace tzdata 2010h-0ubuntu0.9.10 (using .../tzdata_2010i-0ubuntu0.9.10_all
.deb) ...
Unpacking replacement tzdata ...
Setting up tzdata (2010i-0ubuntu0.9.10) ...

Current default time zone: 'America/New_York'
Local time is now:      Mon May  3 16:11:57 EDT 2010.
Universal Time is now:  Mon May  3 20:11:57 UTC 2010.
Run 'dpkg-reconfigure tzdata' if you wish to change it.
```

Updating all packages or upgrading to a new distribution

Rather than updating individual packages, you can update all packages on your system using the `apt-get upgrade` command. Similarly, `apt-get dist-upgrade` will help you migrate to a new level of your distribution.

For more information on other capabilities and options for `apt-get`, see the man page.

APT configuration—the apt.conf file

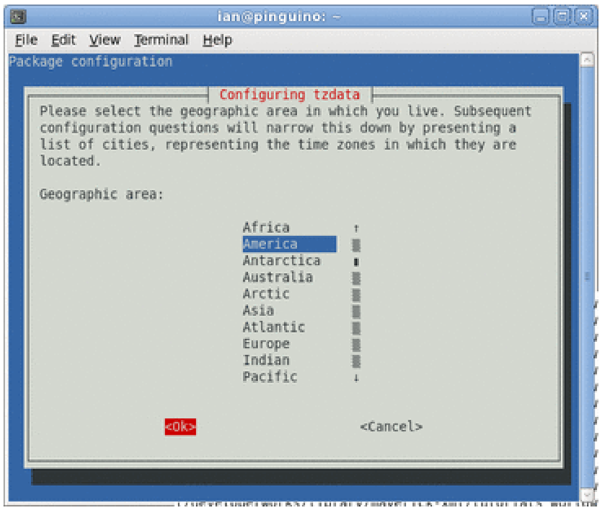
If you check the man page for `apt-get`, you will find that there are many options. If you use the `apt-get` command a lot and find the default options are not to your liking, you can set new defaults in /etc/apt/apt.conf. A program, `apt-config`, is available for scripts to interrogate the apt.conf file. See the man pages for apt.conf and apt-config for further information.

Reconfiguring Debian packages

APT includes a capability called *debconf*, which is used to configure packages after they are installed. Packages that use this capability (and not all do) can be reconfigured after they are installed. The easiest way to do this is to use the `dpkg-reconfigure` command. For example, the `adduser` command may create home directories that are readable by all system users. You may not want this for privacy reasons. Similarly, the tzdata package that we upgraded in Listing 7 suggested running `dpkg-reconfigure tzdata` to change the time zone. You must run `dpkg-reconfigure` with root authority.

Figure 1 shows the first question you will be asked if you run `dpkg-reconfigure tzdata`. The preset default may not be America; it will reflect your own system. Navigate around this text-mode screen using the **Tab** key and the cursor movement keys.

Figure 1. Using dpkg-reconfigure to reconfigure time zone



```
Package configuration

                 Configuring tzdata
 Please select the geographic area in which you live. Subsequent
 configuration questions will narrow this down by presenting a
 list of cities, representing the time zones in which they are
 located.

 Geographic area:

                          Africa
                          America
                          Antarctica
                          Australia
                          Arctic
                          Asia
                          Atlantic
                          Europe
                          Indian
                          Pacific


             <Ok>                      <Cancel>
```

Debian package information

Now let's look at some tools for getting information about packages. Some of these tools do other things as well, but we'll focus here on how to get information.

Package status with dpkg

Another tool that is part of the APT system is the `dpkg` tool. This is a medium-level package management tool that can install and remove packages as well as display status information. Configuration of dpkg can be controlled by /etc/dpkg/dpkg.cfg, and you may also have a .dpkg.cfg file in your home directory to provide further configuration.

The dpkg tool uses many files in the /var/lib/dpkg tree in your filesystem. In particular, the file /var/lib/dpkg/status contains status information about packages on your system. Listing 8 shows the use of `dpkg -s` to display the status of the tzdata package after we updated it and the gcl package after we removed it. Note that the gcl package still shows some config files as remaining. You can also use the `purge` option to purge downloaded package files from the cache and remove configuration information.

Listing 8. Tzdata package status

```
ian@pinguino:~$ dpkg -s gcl tzdata
Package: gcl
Status: deinstall ok config-files
Priority: optional
Section: interpreters
Installed-Size: 152848
Maintainer: Ubuntu MOTU Developers <ubuntu-motu@lists.ubuntu.com>
Architecture: i386
Version: 2.6.7-45ubuntu1
Config-Version: 2.6.7-45ubuntu1
Depends: libc6 (>= 2.7), libgmp3c2, libice6 (>= 1:1.0.0), libncurses5 (>= 5.6+20071006-3
), libreadline5 (>= 5.2), libsm6, libx11-6, libxaw7, libxext6, libxmu6, libxt6, tcl8.4 (
>= 8.4.16), tk8.4 (>= 8.4.16), debconf (>= 1.2.0), gcc, emacs22 | emacsen
Suggests: gcl-doc
Conffiles:
 /etc/default/gcl 9301be50652f86b8d3f8b835f6dce03e
 /etc/emacs/site-start.d/50gcl.el 12116c8c8988326764799973a0a7d5ab
Description: GNU Common Lisp compiler
 GNU Common Lisp (GCL) is a Common Lisp compiler and interpreter
 implemented in C, and complying mostly with the standard set
 forth in the book "Common Lisp, the Language I".  It attempts
 to strike a useful middle ground in performance and portability
 from its design around C.
 .
 This package contains the Lisp system itself.  Documentation
 is provided in the gcl-doc package.
Original-Maintainer: Camm Maguire <camm@enhanced.com>

Package: tzdata
Status: install ok installed
Priority: required
Section: libs
Installed-Size: 6276
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Architecture: all
Version: 2010i-0ubuntu0.9.10
Replaces: libc0.1, libc0.3, libc6, libc6.1
Provides: tzdata-squeeze
Depends: debconf (>= 0.5) | debconf-2.0
Description: time zone and daylight-saving time data
 This package contains data required for the implementation of
 standard local time for many representative locations around the
 globe. It is updated periodically to reflect changes made by
 political bodies to time zone boundaries, UTC offsets, and
 daylight-saving rules.
```

```
Original-Maintainer: GNU Libc Maintainers <debian-glibc@lists.debian.org>
```

Packages and the files in them

You will often want to know what is in a package or what package a particular file came from. These are both tasks for dpkg. Listing 9 illustrates the use of `dpkg -L` to list the files (including directories) installed by the libparted package. For most packages, you can just give the package name and not worry about specifying a particular version. However, some packages may be available in multiple versions, so you may need to specify a more detailed package name when using `dpkg` to interrogate the package information.

Listing 9. What is in the libparted package?

```
ian@pinguino:~$ dpkg -L libparted
Package `libparted' is not installed.
Use dpkg --info (= dpkg-deb --info) to examine archive files,
and dpkg --contents (= dpkg-deb --contents) to list their contents.
ian@pinguino:~$ dpkg -L libparted1.8-12
/.
/lib
/lib/libparted-1.8.so.12.0.0
/usr
/usr/share
/usr/share/doc
/usr/share/doc/libparted1.8-12
/usr/share/doc/libparted1.8-12/copyright
/usr/share/doc/libparted1.8-12/changelog.Debian.gz
/lib/libparted-1.8.so.12
```

To find which package contains a specific file, use the `-S` option of dpkg, as shown in Listing 10. The name of the package is listed on the left.

Listing 10. What package contains a file?

```
ian@pinguino:~$ dpkg -S /lib/libparted-1.8.so.12
libparted1.8-12: /lib/libparted-1.8.so.12
```

Sometimes a file may not appear to belong to any package. When this occurs, you may need to do some extra sleuthing to find where a package comes from. For example, the installation setup step may perform tasks such as creating symbolic links that are not listed as part of the package contents. A relatively recent addition to Linux systems is the *alternatives* system, which is managed using the `update-alternatives` command. Alternatives are frequently created for commands such as `java`, which might be the openJDK, Sun or IBM version, among other possibilities.

Listing 11 shows how to use the `which` command to find what is invoked if we try to run `java`. Then we use the `ls` command to see what that `java` command is symbolically linked to. The link to the /etc/alternatives directory is a tip-off that we are using the alternatives system, so we use the `update-alternatives` command to find more information and finally we use the `dpkg -S` command to confirm that the `java` command comes from the openjdk-6-jre-headless. The setup for the alternatives system would have been done by a post-install script that is part of the openjdk-6-jre-headless package.

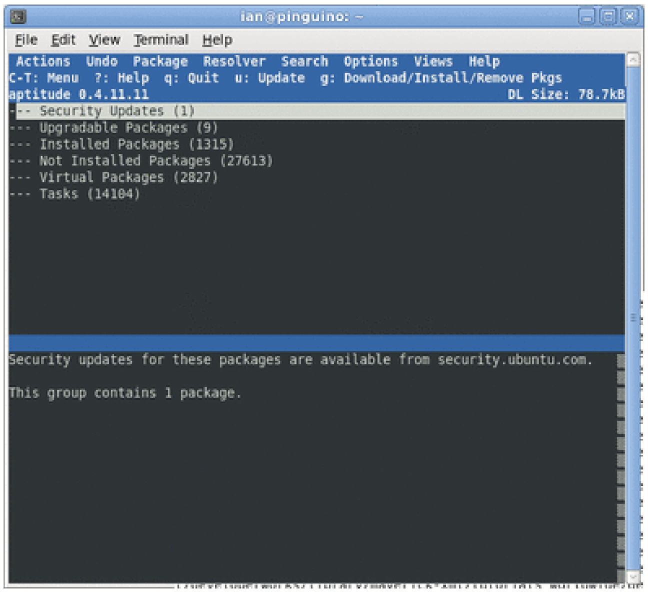
Listing 11. A more complex use of dpkg -S

```
ian@pinguino:~$ which java
/usr/bin/java
ian@pinguino:~$ ls -l $(which java)
lrwxrwxrwx 1 root root 22 2010-05-03 17:51 /usr/bin/java -> /etc/alternatives/java
ian@pinguino:~$ update-alternatives --display java
java - auto mode
 link currently points to /usr/lib/jvm/java-6-openjdk/jre/bin/java
/usr/lib/jvm/java-6-openjdk/jre/bin/java - priority 1061
 slave java.1.gz: /usr/lib/jvm/java-6-openjdk/jre/man/man1/java.1.gz
Current `best' version is /usr/lib/jvm/java-6-openjdk/jre/bin/java.
ian@pinguino:~$ dpkg -S /usr/lib/jvm/java-6-openjdk/jre/bin/java
openjdk-6-jre-headless: /usr/lib/jvm/java-6-openjdk/jre/bin/java
```

Using aptitude

Earlier, we mentioned that the status for packages is kept in /var/lib/dpkg/status. We also mentioned that dpkg could do more than just display package information. Now let's look at the `aptitude` command, which provides a text-based full-screen interface (using ncurses) to the APT package management functions. You can use aptitude to install or remove packages as well as to control status flags that indicate whether packages should be kept up-to-date or held in their present state, for example. If you run the `aptitude` command (as root), you see a screen similar to Figure 2.

Figure 2. Running aptitude



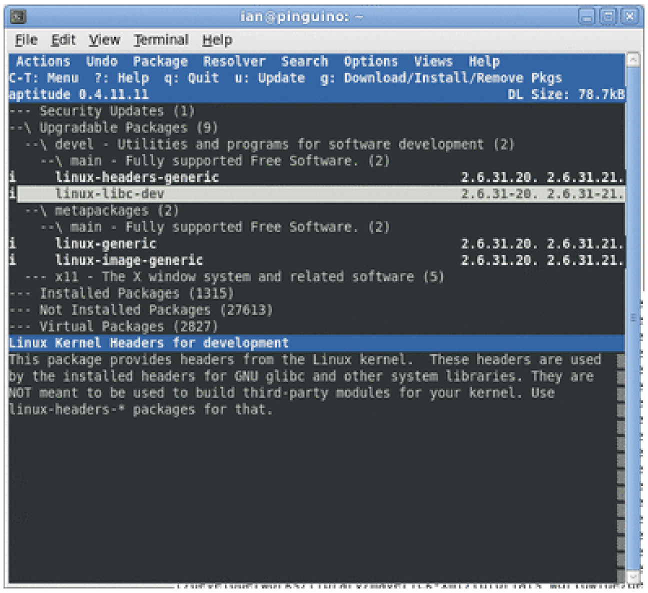
```
 Actions  Undo  Package  Resolver  Search  Options  Views  Help
C-T: Menu  ?: Help  q: Quit  u: Update  g: Download/Install/Remove Pkgs
aptitude 0.4.11.11                                     DL Size: 78.7kB
--- Security Updates (1)
--- Upgradable Packages (9)
--- Installed Packages (1315)
--- Not Installed Packages (27613)
--- Virtual Packages (2827)
--- Tasks (14104)

Security updates for these packages are available from security.ubuntu.com.

This group contains 1 package.
```

Press **Enter** to expand or collapse the various selections, then use **ctrl-t** to access the menu bar. Figure 3 shows that a new kernel version, 2.6.31.20, is available for my system, among other available updates. The 'i' in the left column indicates that the current status is to install the package. The Help menu item explains the various options you have, including holding a package at its current level rather than updating it, removing it, or marking it as being automatically installed and thus eligible for automatic removal. Remember the `autoremove` option of `apt-get`? Now you know how to examine or control which packages are eligible for automatic removal. Use the keyboard shortcuts described in the **Help** or use the **Package** menu item to change the flags.

Figure 3. Running aptitude and examining package flags



```
 Actions  Undo  Package  Resolver  Search  Options  Views  Help
C-T: Menu  ?: Help  q: Quit  u: Update  g: Download/Install/Remove Pkgs
aptitude 0.4.11.11                                      DL Size: 78.7kB
--- Security Updates (1)
--\ Upgradable Packages (9)
  --\ devel - Utilities and programs for software development (2)
    --\ main - Fully supported Free Software. (2)
i     linux-headers-generic                        2.6.31.20. 2.6.31.21.
i     linux-libc-dev                               2.6.31-20. 2.6.31-21.
  --\ metapackages (2)
    --\ main - Fully supported Free Software. (2)
i     linux-generic                                2.6.31.20. 2.6.31.21.
i     linux-image-generic                          2.6.31.20. 2.6.31.21.
  --- x11 - The X window system and related software (5)
--- Installed Packages (1315)
--- Not Installed Packages (27613)
--- Virtual Packages (2827)
Linux Kernel Headers for development
This package provides headers from the Linux kernel.  These headers are used
by the installed headers for GNU glibc and other system libraries. They are
NOT meant to be used to build third-party modules for your kernel. Use
linux-headers-* packages for that.
```

You can use the slash ("/") key to search for packages. For example, if you wanted to reinstall the gcl package that we removed earlier, simply type "/gcl" to search for it. If the search takes you to something else with gcl in it, such as gcl-doc, press the "n" key to advance to the next match. Then use the **Package** menu to mark the package for installation.

When you are finished, select **Actions->Install/remove packages** (or press "g") to apply your selections to the system. You can also select the quit option if you do not want to apply the changes.

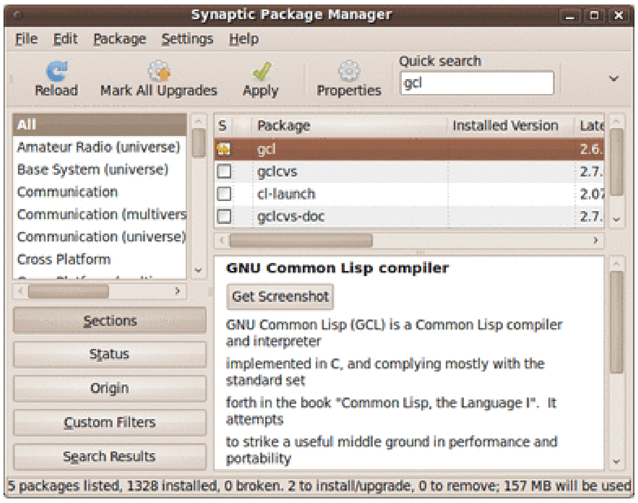
For help at any time, use the menu bar or type "?" (question mark) for help, and then press the "q" key to exit the help.

Upgrading Debian with other tools

You have now seen that aptitude can help you install or remove individual packages as well as upgrade all the packages on your system to the latest level.

In addition to aptitude, there are several other interactive package management interfaces for Debian systems, including dselect, synaptic, update-manager, gnome-apt, and wajig. Synaptic is a graphical application for use with the X Window System. Figure 4 shows the synaptic user interface with our old friend, the gcl package, marked for installation.

Figure 4. Installing gcl using synaptic



Quick search: gcl

All
Amateur Radio (universe)
Base System (universe)
Communication
Communication (multivers
Communication (universe)
Cross Platform

S	Package	Installed Version	Late
	gcl		2.6.
☐	gclcvs		2.7.
☐	cl-launch		2.07
☐	gclcvs-doc		2.7.

GNU Common Lisp compiler

GNU Common Lisp (GCL) is a Common Lisp compiler and interpreter

implemented in C, and complying mostly with the standard set

forth in the book "Common Lisp, the Language I". It attempts

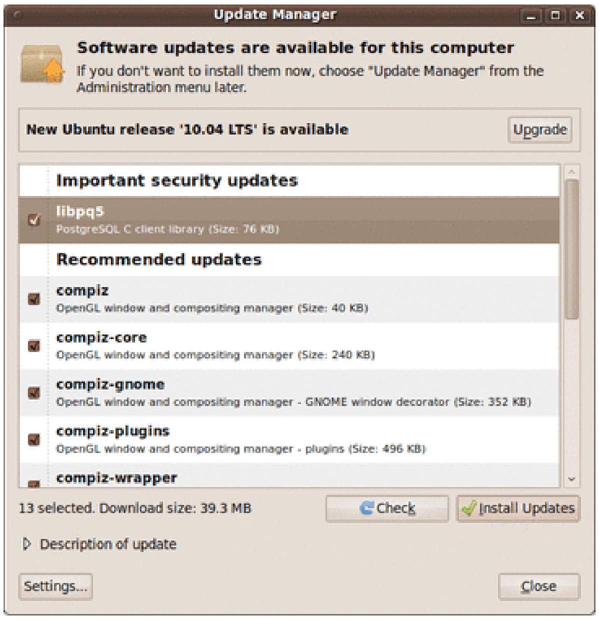
to strike a useful middle ground in performance and portability

5 packages listed, 1328 installed, 0 broken. 2 to install/upgrade, 0 to remove; 157 MB will be used

The **Apply** button will install gcl and update any other packages that are scheduled for update. The **Reload** button will refresh the package lists. If you are accustomed to GUI interfaces, you may find synaptic easier to use than apt-get, dpkg, or dselect.

Similarly, you may find that your system includes `update-manager`, an X Window System application, specifically tailored to help you keep your system up-to-date. If installed, it is likely to be started automatically on a regular basis, so that you don't forget to update. Figure 5 shows how update manager displays the set of updates that you saw in Figure 2. As was the case with aptitude, the updates are classified so you know which updates are important security updates.

Figure 5. Update Manager example



Software updates are available for this computer

If you don't want to install them now, choose "Update Manager" from the Administration menu later.

New Ubuntu release '10.04 LTS' is available

Important security updates

✓ **libpq5** PostgreSQL C client library (Size: 76 KB)

Recommended updates

✓ **compiz** OpenGL window and compositing manager (Size: 40 KB)

✓ **compiz-core** OpenGL window and compositing manager (Size: 240 KB)

✓ **compiz-gnome** OpenGL window and compositing manager - GNOME window decorator (Size: 352 KB)

✓ **compiz-plugins** OpenGL window and compositing manager - plugins (Size: 496 KB)

✓ **compiz-wrapper**

13 selected. Download size: 39.3 MB

Finding Debian packages

In our final topic on Debian package management, we look at ways to find packages. Usually, apt-get and the other tools discussed here will already know about any Debian package you might need from the list of available packages. A command that we haven't used yet is `apt-cache`, which is useful for searching package information on your system. `apt-cache` can search using regular expressions (see Learn Linux, 101: Search text files using regular expressions for more information on regular expressions). Suppose you wanted to find the name of the package containing the Linux loader. Listing 12 shows how you might accomplish this.

Listing 12. Searching for the Linux loader with apt-cache

```
ian@pinguino:~$ apt-cache search "linux loader"
lilo - LInux LOader - The Classic OS loader can load Linux and others
lilo-doc - Documentation for LILO (LInux LOader)
```

You saw earlier that aptitude and synaptic also offer search tools. If you use synaptic, note that you have options on the search menu for searching only package names or package descriptions as well.

If you still can't find the package, you may be able to find it among the list of packages on the Debian site (see Resources for a link), or elsewhere on the Internet.

Most of the package tools can tell you a lot more about an installed package than about one that you do not yet have installed, such as the list of files within a package. If you need to find what package contains a program that you do not have installed, there are a few ways:

- You can guess what package might contain it and download the package without installing. Once you have the package, you can interrogate it.
- You can search the Internet.
- You can try the command-not-found capability, which is described under "Command not found" later in this article.

The `apt-get` command has a `-d` option to download a package and not install it. There is also a `--print-uris` option to show where a package would be downloaded from and what its checksum would be. Current checksums are likely to be SHA256 checksums, so you can check the integrity of the downloaded package using the `sha256sum` command. Note that the URI and checksum information are not displayed if you have already downloaded the package, so you should get this information before downloading the package.

Suppose you want to know whether the gcl command is really contained in the gcl package. Listing 13 shows you how to use `apt-get` to just download the gcl package without installing it.

Listing 13. Using apt-get for download-only

```
ian@pinguino:~$ sudo apt-get install -d gclReading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libreadline5
Suggested packages:
  gcl-doc
The following NEW packages will be installed:
  gcl libreadline5
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 47.1MB of archives.
After this operation, 157MB of additional disk space will be used.
Do you want to continue [Y/n]?
Get:1 http://us.archive.ubuntu.com karmic/main libreadline5 5.2-6 [140kB]
Get:2 http://us.archive.ubuntu.com karmic/universe gcl 2.6.7-45ubuntu1 [47.0MB]
Fetched 47.1MB in 7s (6,475kB/s)
Download complete and in download only mode
```

Once you have the package downloaded, you can use the `--info` option of `dpkg` to display the package information, or the `--contents` option to show what files the package installs. The downloaded file will be usually be located in /var/cache/apt/archives/. Listing 14 shows how to locate the file you downloaded and find out what binaries it will install (assuming they are installed in a .../bin/... directory).

Listing 14. Using dpkg to list the contents of a .deb

```
ian@pinguino:~$ sudo find /var/cache -name "*.deb"
/var/cache/apt/archives/gcl_2.6.7-45ubuntu1_i386.deb
/var/cache/apt/archives/libreadline5_5.2-6_i386.deb
ian@pinguino:~$ sudo dpkg --contents /var/cache/apt/archives/gcl*.deb| grep "/bin/"
drwxr-xr-x root/root         0 2008-12-06 07:42 ./usr/bin/
-rwxr-xr-x root/root       617 2008-12-06 07:42 ./usr/bin/gcl
```

If you do find and download a `.deb` file using something other than `apt-get`, you can install it using `dpkg -i`.

If you decide you do not want to install the package you downloaded into the APT archives, you can run `apt-get clean` to clear out any downloaded package files.

If all else fails, there is another possible source for packages. Suppose you find a program packaged as an RPM rather than a .deb. You may be able to use the `alien` program, which can convert between package formats. You should read the alien documentation carefully as not all features of all package management systems can be converted to another format by alien.

Command not found

Back in Listing 1, you saw a helpful message telling you what package to install to get the `gcl` command. How is this done? When the Bash shell searches for a command and does not find it, then the shell searches for a shell function named `command_not_found_handle`. Listing 15 shows how this is defined on my Ubuntu 9.10 system.

Listing 15. command_not_found_handle

```
ian@pinguino:~$ type command_not_found_handle
command_not_found_handle is a function
command_not_found_handle ()
{
    if [ -x /usr/lib/command-not-found ]; then
        /usr/bin/python /usr/lib/command-not-found -- $1;
        return $?;
    else
        return 127;
    fi
}
```

If the `command_not_found_handle` function exists, it is invoked with the original command and original arguments as its arguments, and the function's exit status becomes the exit status of the shell. If the function is not defined, the shell prints an error message and returns an exit status of 127. The function is usually set in the system /etc/bash.bashrc file. You can see from Listing 15 that the function checks for the existence of /usr/lib/command-not-found and runs it as a Python script if it exists. If it does not exist, perhaps because the command-not-found package that supplies it was removed after the shell session was started, then the function mimics the standard system behavior and returns 127.

PackageKit

No discussion of package installation would be complete without mentioning PackageKit, which is a system designed to make installing and updating software easier. The intent is to unify all the software graphical tools used in different distributions. PackageKit uses a system activated daemon, which means that the daemon is activated only when needed. Packagekit has version for Gnome (gnome-packagekit) and KDE KPackageKit).

There is a lot more to the Debian package management system than covered here. There is also a lot more to Debian than its package management system. See Resources for additional details and links to other articles in this series.

Resources

Learn

- Use the developerWorks roadmap for LPIC-1 to find the developerWorks articles to help you study for LPIC-1 certification based on the April 2009 objectives.
- At the LPIC Program site, find detailed objectives, task lists, and sample questions for the three levels of the Linux Professional Institute's Linux system administration certification. In particular, see their April 2009 objectives for LPI exam 101 and LPI exam 102. Always refer to the LPIC Program site for the latest objectives.
- Review the entire LPI exam prep series on developerWorks to learn Linux fundamentals and prepare for system administrator certification based on earlier LPI exam objectives prior to April 2009.
- The Debian home page is your starting point for information about the Debian distribution.
- Refer to the "Debian installation guide" for more information on installing Debian.
- At Debian packages, display and search Debian package lists.
- See the "APT HOWTO" to learn more about the Debian package management utility, APT.
- Check out the "Debian New Maintainers' Guide."
- Visit Ubuntu, Linux for Human Beings.
- See the PackageKit home page to learn more about PackageKit.
- The Linux Documentation Project has a variety of useful documents, especially its HOWTOs.
- In the developerWorks Linux zone, find hundreds of how-to articles and tutorials, as well as downloads, discussion forums, and a wealth other resources for Linux developers and administrators.
- Stay current with developerWorks technical events and webcasts focused on a variety of IBM products and IT industry topics.
- Attend a free developerWorks Live! briefing to get up-to-speed quickly on IBM products and tools as well as IT industry trends.
- Watch developerWorks on-demand demos ranging from product installation and setup demos for beginners, to advanced functionality for experienced developers.
- Follow developerWorks on Twitter, or subscribe to a feed of Linux tweets on developerWorks.

Get products and technologies

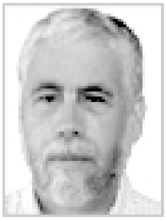
- Get the Alien package converter.
- Get Debian packages from the Debian home page.
- Evaluate IBM products in the way that suits you best: Download a product trial, try a product online, use a product in a cloud environment, or spend a few hours in the SOA Sandbox learning how to implement Service Oriented Architecture efficiently.

Discuss

- Participate in the discussion forum for this content.
- Get involved in the My developerWorks community. Connect with other developerWorks users while exploring the developer-driven blogs, forums, groups, and wikis.

About the author

Ian Shields



Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents and has published several papers. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. Learn more about Ian in Ian's profile on developerWorks Community.