Installing Debian Software with the Advanced Package Tool

Will Trillich

wdt@users.sourceforge.net

This is an introduction to Debian powerful Advanced Package Tool to install new packages.

SGML source for this document is available on the Newbiedoc Project (newbiedoc.sourceforge.net)'s site.

1. Introduction

1.1. About APT

Once your Debian system is running, you can add new features and facilities to it by installing software PACKAGES -- and you'll seldom, if ever, have to restart!

Debian uses a cool packaging system "APT" that takes care of dependencies for you -- **package_A** may require **package_B**, which requires **package_C**, and the Advanced Package Tool will take care of installing **package_C** and **package_B** if you request **package_A**. Very nice!

It's called **APT** as in Advanced Package Tool.

1.2. Using APT with the shell

To see which **apt-*** commands you have available, you can use a handy feature that's offered by most command shells. (A command shell is the program that interprets your commands and executes your instructions, such as when you type **cd** or **ls** or **ps**. In fact, to find which command shell you're using, that last command will do the trick: **ps** will list all your processes, including **ps** itself and your command shell, which is probably **bash**)

So, try this at your shell's command prompt: type '**apt**', and INSTEAD OF pressing **ENTER** , try **TAB** instead:

```
# apt <TAB KEY>
```

Most command shells will then display for you all the commands that start with those keystrokes. It's a great way to find useful nuggets!

This feature is called 'completion', and most command shells offer it in some fashion.

You can find more details about it in the manpage for your shell: **man tcsh**,**man bash**, and maybe others. If you have to press tab twice to get a listing, compare your list with mine below, and then check the manpage and see if you can figure out why it's different... Plus, if completion is not working in your shell, either you or your sysadmin probably turned a feature off. Again, the manpage will help you here.

To see the manual on just about any command, use **man**, as in:

```
% man apt-get
% man exim.conf
% man egrep
```

See why we call them manpages? :)

On my system, using the **tcsh** shell, here's what I get:

```
%  apt <^D>
apt-cache  apt-cdrom  apt-config apt-get
apt-move   apt-setup  aptitude
%
```

That is just a snapshot of my system, which is likely to change when I install something else later on this afternoon... (with **apt**, it's so easy to install stuff, it's nearly dangerous!)

Your system will probably vary, depending on which shell you're using and which packages you have installed, of course.

I'll only talk about **apt-setup, apt-get** and **apt-cache** here, plus a bit of a related program, **dpkg**.

2. Using APT to install packages

2.1. Assumptions

I use the POTATO (version 2.2) release of Debian in my discussion here. To see which version you've got, try

```
% uname -r
```

My system is version '2.2.17'.

The **uname** utility is available in the **shellutils** package:

```
# apt-get install shellutils
```

to get it, if you don't have it.

In addition, I use the **tcsh** command shell, just to be difficult.

Note that your system is BOUND to be different from mine -- type and amount of memory, drive sizes, peripheral gadgets -- so YMMV (your mileage may vary) wildly from mine. Even so, with luck it might prove useful nonetheless...

2.2. Setup -- *apt-setup*

This little gizmo sets up the configuration file for you:

```
# apt-setup
```

It asks you some questions and winds up configuring your `/etc/apt/sources.list` file. It tells **apt** where to get the packages from, when you request an update to your currently installed packages or when you ask to install packages you've never tried before.

If you don't have **apt-setup** on your potato (or later) Debian system, get the **base-config** package, which supplies it:

```
# apt-get install base-config
```

Of course, you can always set up **apt**'s `sources.list` file by hand if you must. Check out

```
% man sources.list
```

to learn about the format of the 'sources.list' file, if you're the determined type. For a list of Debian mirrors to use there, try http://www.debian.org/misc/README.mirrors" (http://www.debian.org/misc/README.mirrors) But if you're like me -- lazy -- you'll just run

```
# apt-setup
```

and be done with it.

2.3. Following a distribution

Keep in mind that Debian is a work-in-progress, meaning that as holes are found and bugs are killed in the **stable** distribution, a whole new world is developing on the **UNSTABLE** distribution. If you need something from 'unstable' go ahead and try it -- just don't gamble more than you can afford to lose, right? Venturing into **UNSTABLE** is fine if you don't mind being on the "front lines" so to speak. Most of us stick with the **stable** distribution, which has no new gizmos being created; only patches and fixes are added to the **stable** release. You might occasionally want to delve into **UNSTABLE** territory, for some new functionality that's not available in the stodgy, old, stable area. It's up to you, but don't expect a refund if something breaks -- it's called **UNSTABLE** for a reason!

Starting in 2001 a new distribution of Debian is available. It is called **testing**, and it covers the ground between **stable** and **UNSTABLE**. Testing is made of packages that have survived 14 days in unstable without breaking. Major life-threatening bugs are thus solved before making their way into testing. However, that also means that security upgrades are also at least 14 days behind schedule...

However if your version of **apt** supports it (>= 0.5), there is a very easy way to follow multiple distributions, it is called **pinning:**

You must modify `/etc/apt/preferences` and add:

```
Package: *
Pin: release a=stable
Pin-Priority: 900

Package: *
Pin: release a=testing
Pin-Priority: -10
```

then you must add lines for *both* stable *and* testing to your `/etc/apt/sources.list` and do an **apt-get update** which will download the usual files *twice*, one for each distribution.

After this, you can use the **-t** option to choose which distribution you want to get packages from:

```
# apt-get -t testing install sgmltools2
```

The **Pin-Priority** fields ensure that unless you specifiy it manually, all packages will be taken from the stable distribution (of course, dependencies are always met, so you might have to download more than one package from testing)

If you're running a live server, where any uncertainty or instability would be a definite liability, *definitely* shy away from **UNSTABLE** or even **testing** unless you enjoy soothing the frazzled nerves of management, and their paying customers, and your spouse, who keeps asking why you're always looking for a new job.

2.4. Updating your system -- *apt-get upgrade*

So you've got your Debian system humming -- you don't need to let it fall behind! You can easily stay up-to-date with the latest security patches, any bug fixes and an occasional enhancement. That's the beauty of **apt**.

Once `sources.list` is set up and you're online, first

```
# apt-get update
```

to refresh the list of available packages -- if new things are available, that is how your Debian system will learn of it -- and then simply

```
# apt-get upgrade
```

which downloads, configures and then replaces any packages you've already got that have been tweaked. You'll probably want to do it periodically, to squash bugs and plug security holes.

" NOTE that the Debian gurus work like the dickens to see to it that your *settings are not clobbered* when you upgrade a package or two. You may have spent a month getting **exim** or **inetd** to hum your very own tune, and heaven help the miscreant who clobbers your settings, right? Sometimes there's enough of a difference between what you've been running (the old version) and the upgrade to be installed (the new version) that the old settings may not be usable by the new program; in this unusual case, **apt** informs you that *your old settings are being saved,* and in order for the new program to function the way you want it to, you'll have to migrate your settings by hand. But this is rare -- usually, your settings stay right where you put them and you'd hardly ever know anything had improved. "

All this happens without having to restart. (For Mac and Windows people -- imagine leaving your computer up for two MONTHS! It's awesome!)

2.5. Installing packages -- *apt-get install*

Once you're online, it's easy to install new packages you're interested in; simply do something like

```
# apt-get install vim traceroute mysql-server mysql-client
# apt-get install logcheck
```

to download, configure and install whichever packages you want. " Where does **apt** put all this stuff it downloads? The "cache" is located under `/var/cache/apt/`; there's a 'table-of-contents' file there called `pkgcache.bin` (and if you're set up to also get source-code packages, you'll also have `srcpkgcache.bin`) and the actual ***.deb** package files are in `/var/cache/apt/archives/`. "

3. Getting information about packages

3.1. Finding packages -- *apt-cache search*

Whether you're online or not--

How do you find the package that's got the feature you're looking for? First, do

```
# apt-get update
```

so your package list is up-to-date, and then try something like

```
% apt-cache search tunnel
% apt-cache search 'php.*sql'
% apt-cache search apache.\*perl
```

```
% apt-cache search elvis\|vim
```

That is how you tell **apt** to search the packages you've downloaded, using REGEX (regular expression, a pattern-matching 'language') -- if your pattern uses any keystrokes that mean something to your command shell (e.g. **[|?*]**) you'll need to quote them so that **apt-cache** will be able to see them, instead of having the shell expand the term to a list of file names that mean something else entirely.

" NOTE -- **apt-cache** only knows about the package descriptions you've already downloaded. To search among ALL known Debian packages just browse to **http://packages.debian.org/PACKAGESUBSTRING** to see what's available. For example: http://packages.debian.org/vnc (http://packages.debian.org/vnc) That would get you a listing of packages that contain the term "vnc" somewhere in the title."

3.2. Describing packages -- *apt-cache show*

The simplest way to see the description of a package:

```
% apt-cache show postgresql
% apt-cache show iproute
```

That displays what the package is designed to do, version info and so forth. (Pretty simple, huh? Not everything worthwhile needs to be complicated...)

3.3. Listing packages -- *dpkg -l*

Which packages are installed? Do any packages need configuring?

```
% dpkg -l
```

(that's a lower-case EL, not a one.) That lists all INSTALLED packages.

```
% dpkg -l \*
% dpkg -l '*'
```

These list all packages. (Without the star glob, **dpkg** only lists installed packages.)

```
% dpkg -l \*postgres\*
% dpkg -l '*postgres*'
```

That shows the status of packages matching that GLOB (it's not a regular expression [REGEX]: a regex interprets **.*** to mean '"." = any character, "*" = zero or more times'; a glob interprets **.*** to mean '"." = a dot, "*" = followed by zero or more characters').

NOTE: If your pattern uses fancy characters that have a special meaning to your command shell (*** ? |** etc.) you'll need to QUOTE them so that your command shell doesn't interpret them -- you want **dpkg** to see the pattern, instead. Quote such characters by either enclosing them in quotes:

```
'*like|this?or that'
"even[this|is]acceptable"
```

or precede them with the BACKSLASH:

```
this\ acts\ like\ one\ word\ cuz\ spaces\ are\ quoted
\[one\|two\?three\*four\]
```

And note that the /SLASH/ (forward slash) leans right, in the same direction as the text you're reading, whereas the \BACKSLASH\ leans to the left. SLASH is used to delineate components (folders and subfolders, if you like) along a directory path (`/home/will/public_html/index.html` for example); BACKSLASH is used to alter the meaning of the keystroke that follows. Very important distinction! (And curses to the weenie who first thought QDOS -- which became MS-DOS -- should use backslashes as path delimiters! Bad dog!)

3.4. Combine *dpkg* with *grep* for some powerful searches

```
% dpkg -l \* | grep ^pi
```

finds installed packages marked to be purged.

```
% dpkg -l '*' | grep "^i[^i]"
```

lists packages marked for installation, that aren't installed yet.

```
% dpkg -l \* | grep '^[^i]i'
```

shows installed packages that are marked for anything but installation (i.e.uninstallation or purgery). See **man grep** for more info on **grep** and **man dpkg** for more on the listing format.

On my system, to find out which **apt** packages I have installed, I can do this:

```
%  dpkg -l \*apt\*
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name              Version           Description
+++-=================-=================-==================================================
ii  apt               0.3.19            Advanced front-end for dpkg
ii  apt-move          3.0-13            Move cache of Debian packages into a mirror hierarch
ii  aptitude          0.0.4a-4.1        Console based apt frontend
un  libapt-pkg-dev    <none>            (no description available)
un  libapt-pkg-doc    <none>            (no description available)
un  libapt-pkg2.7     <none>            (no description available)
pn  task-laptop       <none>            (no description available)
```

Here, **dpkg** displays its headers, and then shows all packages that have "apt" anywhere in the title. The three **libapt-*** packages are not installed (I've **hilighted** the "apt" strings that belong to non-**apt** packages, so you can see why they showed up. Interesting, no?)

For a slightly different approach, how about this:

```
%  dpkg -l \* | grep apt
ii  apt               0.3.19            Advanced front-end for dpkg
ii  apt-move          3.0-13            Move cache of Debian packages into a mirror hierarch
```

```
ii  aptitude          0.0.4a-4.1         Console based apt frontend
un  libapt-pkg-dev    <none>             (no description available)
un  libapt-pkg-doc    <none>             (no description available)
un  libapt-pkg2.7     <none>             (no description available)
ii  libpcap0          0.4a6-3            System interface for user-level packet capture.
pn  task-laptop       <none>             (no description available)
%
```

Here I asked to see ALL packages (**dpkg -l ***) and thenI used **grep** to display only those lines from the output that contained "apt". The headers disappeared, since none of them contain "apt" -- so **grep** discards them.

3.5. Package contents -- *dpkg -L*

```
% dpkg -L mysql
% dpkg -L apt
```

(with an upper-case EL) shows all the files -- with full path names -- that are provided by the package.

3.6. Finding which package contains... -- *dpkg -S*

How can you determine here did file XYZ come from? Here’s how to find which package contains/supplies a certain file:

```
% dpkg -S postmaster
% dpkg -S 'doc/*sql' | cut -f1 -d: | sort -u
```

That searches for packages that supply files whose paths contain the requested GLOB. See **man cut** and **man sort** for info on how to use these tools in your day-to-day fiddling.

If you’re using **unstable** then you might be able to tinker with **dlocate** as well. I hear it’s highly thought-of by all who’ve used it. (If you know about it and want to include some info here, please let me know!)

By the way -- here’s a quickie command to show what packages I’ve got installed that provided all of the **apt-*** files you saw above when I tried **`apt<^D>`** :

```
%  dpkg -S apt- | sort
apt-move: /etc/apt-move.conf
apt-move: /etc/cron.weekly/apt-move
apt-move: /usr/bin/apt-move
apt-move: /usr/share/doc/apt-move
apt-move: /usr/share/doc/apt-move/README.Debian
apt-move: /usr/share/doc/apt-move/README.gz
apt-move: /usr/share/doc/apt-move/TODO
apt-move: /usr/share/doc/apt-move/TODO.Debian
apt-move: /usr/share/doc/apt-move/changelog.Debian.gz
apt-move: /usr/share/doc/apt-move/changelog.gz
apt-move: /usr/share/doc/apt-move/copyright
apt-move: /usr/share/doc/apt-move/examples
apt-move: /usr/share/doc/apt-move/examples/SAMPLE.exclude
```

```
apt-move: /usr/share/man/man8/apt-move.8.gz
apt: /usr/bin/apt-cache
apt: /usr/bin/apt-cdrom
apt: /usr/bin/apt-config
apt: /usr/bin/apt-get
apt: /usr/lib/libapt-pkg.so.2.7
apt: /usr/lib/libapt-pkg.so.2.7.1
apt: /usr/share/man/man8/apt-cache.8.gz
apt: /usr/share/man/man8/apt-cdrom.8.gz
apt: /usr/share/man/man8/apt-config.8.gz
apt: /usr/share/man/man8/apt-get.8.gz
base-config: /usr/sbin/apt-setup
base-config: /usr/share/debconf/templates/apt-setup.templates
base-config: /usr/share/man/man8/apt-setup.8.gz
```

That shows three of the apt packages I've got installed: **apt, apt-move,** and **base-config.** You'll notice that **aptitude** isn't listed -- which is understandable, since it doesn't contain the "apt DASH" string. Also, **base-config** didn't show up above when I searched via **dpkg -l * | grep apt** since the string "apt" doesn't appear within **base-config**. See?

As is the case with many packages, most of the files are documentation: either `/usr/share/doc/*` or manpages.

NOTE that if I do **dpkg -S apt** without the dash after "apt-" i get more irrelevant stuff, because many packages contain files that have the string "apt" in them, including:

- `/usr/bin/`**`apt`**`itude` it'll be a nice APT front-end when it's done
- `/usr/share/doc/gpm/README.synaptics.gz`
- `/usr/share/doc/postgresql-doc/postgres/libpq-chapter18164.htm`
- `/usr/share/texmf/tex/latex/caption`

...so you must learn to be careful what you search for... you'll find it!

4. Learn more

The "apt" facility goes lots deeper and wider than what I've hinted upon here. I've only scratched the surface!

To get your knees wet, try these:

```
% man apt-get
% man apt-cache
% man sources.list
% man dpkg

% man grep
% man cut
% man sort
```

to learn more.

Also check out the manpage for your command shell,

```
% man bash
% man tcsh
% man ksh
```

Look for your shell's GLOB (***part*** aka filename expansion) and COMPLETION (aka filename completion) features; these are powerful tools that can save your patootie from the sling once you know how to use them...

For further reading, and to learn about Debian or Linux in general, visit these websites often:

- www.debian.org
- www.debianHelp.org
- www.debianPlanet.org

and for Linux-general (not Debian specific) info, try

- search.netscape.com/Computers/Software/Operating_Systems/Unix/Administration
- directory.google.com/Top/Computers/Software/Operating_Systems/Linux/
- www.linuxNewbie.com (www.linuxNewbie.com)
- www.linuxNewbie.org (www.linuxNewbie.org)
- www.linuxPlanet.com (www.linuxPlanet.com)

then when you begin to understand the ways of Linux, see

- slashdot.org (slashdot.org)
- www.rootPrompt.org

Apt sure is handy and powerful stuff, once you get used to it.

Last update: Wed Apr 25 02:05:34 2001

SGML formatting: Tue Jul 23

If you have comments or suggestions about this document, please lemme know.

`<will@dontUthink.com?subject=apt-get-intro>`