Debian Quick Reference

Osamu Aoki <osamu\#at\#debian.org>
'Authors' on page 23

CVS, Thu Jan 18 11:54:29 UTC 2007

Abstract

This Debian Quick Reference (http://qref.sourceforge.net/) is intended to provide a short introduction to the Debian system as a **quick reference**. This is an excerpt of Debian Reference (http://qref.sourceforge.net/).

Copyright Notice

Contents

Chapter 1

Preface

This document originated as a "quick reference" but it grew. Nevertheless, **Keep It Short and Simple** (KISS) is my guiding principle.

1.1 Document conventions

This Debian Quick Reference provides information through short `bash` shell commands.

Reference to:
- a UNIX-style **manual page** is given in the form: `bash(1)`.
- a GNU **TEXINFO page** is given in the form: `info libc`.

1.2 Basics of the Debian distributions

Debian maintains three different distributions simultaneously. These are:

- `stable` — Most useful for a production server since it is only updated with security fixes.
- `testing` — The preferred distribution for a workstation since it contains recent releases of desktop software which have received a bit of testing.
- `unstable` — Cutting edge. The choice of Debian developers.

When packages in `unstable` have no release-critical (RC) bugs filed against them after the first week or so, they are automatically promoted to `testing`.

Debian distributions also have code names. Before Sarge was released in June 2005, the three distributions were Woody (stable), Sarge (testing), and Sid (unstable). After Sarge was released the three distributions were, respectively, Sarge, Etch, and Sid. When Etch is released, the

`stable` and `unstable` distributions will be Etch and Sid; a new `testing` distribution will then be created (initially as a copy of `stable`) and given a new code name.

Subscribe to the low-volume mailing list `debian-devel-announce@lists.debian.org` for important announcements about Debian.

If you want to use versions of packages that are more current than the versions that were released with the distribution you are using, then you can either upgrade to a later distribution as described in 'Upgrading a distribution to `stable`, `testing`, or `unstable`' on the next page, or you can upgrade only selected packages. If the package can't be upgraded easily then you may want to backport it as described in 'Port a package to the `stable` system' on page 18.

Chapter 2

Upgrading a distribution to `stable`, `testing`, or `unstable`

2.1 Upgrading from Potato to Woody

This procedure is described separately because Potato's APT did not have all the features described in the current `apt_preferences(5)` manpage.

After including only Woody sources in `/etc/apt/sources.list`, upgrade APT and required core packages to Woody versions by doing the following:

```
# apt-get update
# apt-get install libc6 perl libdb2 debconf
# apt-get install apt apt-utils dselect dpkg
```

Then upgrade the rest of the system to Woody.

```
# apt-get upgrade
# apt-get dist-upgrade
```

2.2 Preparing for upgrade

You can upgrade from one distribution to another one by fetching packages over the network. This can be done as follows.

Get a clean list of repositories for `stable`:

```
# cd /etc/apt
# cp -f sources.list sources.list.old
# :>sources.list
# apt-setup noprobe
```

If you want to upgrade to testing then add testing sources to this new list. If you want to upgrade to unstable then also add unstable sources.

```
# cd /etc/apt
# grep -e "^deb " sources.list >srcs
# :>sources.list
# cp -f srcs sources.list
# sed -e "s/stable/testing/" srcs >>sources.list
# sed -e "s/stable/unstable/" srcs >>sources.list
# apt-get update
# apt-get install apt apt-utils
```

See 'Beginning Debian package management' on page 6 for the art of tuning /etc/apt/sources.list and /etc/apt/preferences.

2.3 Upgrading

After properly setting up /etc/apt/sources.list and /etc/apt/preferences as described above you can begin the upgrade.

Note that tracking the testing distribution of Debian can have the side effect of delaying the installation of packages containing security fixes, since such packages are uploaded to unstable and only later migrate to testing.

See 'Debian package management' on the facing page for the basics, and see 'APT upgrade troubleshooting' on page 11 if you encounter problems.

2.3.1 Using dselect

If a system has many packages which include -dev packages, etc., the following method using dselect is recommended for fine-grained package control.

```
# dselect update  # always do this before upgrade
# dselect select  # select additional packages
```

All your current packages will be selected when dselect starts. dselect may prompt you with additional packages based on Depends, Suggests, and Recommends. If you do not want to add any packages, just type Q to exit dselect again.

```
# dselect install
```

You will have to answer some package configuration questions during this part of the process, so have your notes ready and allow some time for this part. See 'dselect' on page 8.

Use dselect. **It always works :)**

Chapter 3

Debian package management

`aptitude` is now the preferred text front end for APT, the Advanced Package Tool. It remembers which packages you deliberately installed and which packages were pulled in through dependencies; the latter packages are automatically de-installed by `aptitude` when they are no longer needed by any deliberately installed packages. It has advanced package-filtering features but these can be difficult to configure.

`synaptic` is now the preferred Gtk GUI front end for APT. Its package filtering capability is easier to use than `aptitude`'s. It also has experimental support for Debian Package Tags (`http://debtags.alioth.debian.org/`).

To reduce the network load on the Debian repositories and to speed up your downloads you should get packages from Debian mirror sites.

If you need to install the same package on several machines on your local network then you can set up a local HTTP proxy using `squid` for packages downloaded through APT. If necessary, set the `http_proxy` environment variable or set the `http` value in `/etc/apt/apt.conf`.

Although APT's pinning feature, described in `apt_preferences(5)`, is powerful, its effects can be difficult to understand and manage. You should consider it an Advanced Feature.

The use of chroot is desirable for simultaneously securing both system stability and access to the latest versions of software.

This chapter is based on a post-Woody system. Some features may require a Sarge system or later.

3.1 Introduction

If reading all the developer documentation is too much for you, read this chapter first and start enjoying the full power of Debian with `testing/unstable` :-)

3.1.1 Main package management tools

```
dpkg       - Debian package file installer
apt-get    - Command line front end for APT
aptitude   - Advanced text and command line front end for APT
synaptic   - Gtk GUI front end for APT
dselect    - Menu-driven package manager
tasksel    - Task installer
```

These tools aren't all alternatives to one another. For example, `dselect` uses both APT and `dpkg`.

APT uses `/var/lib/apt/lists/*` for tracking available packages while `dpkg` uses `/var/lib/dpkg/available`. If you have installed packages using `aptitude` or other APT front ends and you want to use `dselect` to install packages then the first thing you should do is update `/var/lib/dpkg/available` by selecting `[U]pdate` from `dselect`'s menu (or by running "`dselect update`").

`apt-get` automatically installs all packages upon which a requested package Depends. It does not install the packages that a requested package merely Recommends or Suggests.

`aptitude`, in contrast, can be configured to install packages that a requested package Recommends or Suggests.

`dselect` presents the user with a list of packages that a selected package Recommends or Suggests and allows these to be selected or deselected individually.

3.1.2 Convenience tools

```
dpkg-reconfigure  - reconfigure an already installed package
                    (if it uses debconf)
dpkg-source       - manage source package file
dpkg-buildpackage - automate the building of a package file
apt-cache         - check package archive in local cache
```

3.2 Beginning Debian package management

3.2.1 Set up APT

Set up `sources.list` as described in 'Preparing for upgrade' on page 3. [1]

[1] If you track `testing` or `unstable` you can remove references to `stable` from `/etc/apt/sources.list` and `/etc/apt/preferences` because `testing` starts as a copy of `stable`.

3.2.2 Installing tasks

You can install sets of packages typically required in order to put a Debian system to a certain use. These sets of packages are called “tasks”.

The simplest way to install tasks at the time of initial installation is to use `tasksel`. Note that you must run

```
dselect update
```

before using it.

`aptitude` can also install tasks and is the tool recommended for this purpose. It enables you to deselect individual packages within tasks before proceeding to the installation step.

3.2.3 `aptitude`

`aptitude` is a new menu-driven package installer similar to `dselect` but built from scratch on top of APT. It can be used as an alternative to `apt-get` for most commands. See `aptitude(1)` and `/usr/share/doc/aptitude/README`.

Once you start using `aptitude` it is best to continue using it rather than alternative methods of installing packages; otherwise you lose the advantage of `aptitude` keeping track of which packages you have deliberately installed.

`aptitude` in full screen mode accepts single-key commands which are usually lowercase. Notable key strokes are:

```
Keystroke   Action
F10         Menu
?           Help for keystroke (complete listing)
u           Update package archive information
+           Mark the package to be upgraded or newly installed
-           Mark the package to be removed (keep config)
_           Mark the package to be purged (remove config)
=           Place the package on hold
U           Mark all upgradable packages to be upgraded
g           Download and install selected packages
q           Quit current screen and save changes
x           Quit current screen and discard changes
Enter       View information about a package
C           View a package's changelog
l           Change the limit for the displayed packages
/           Search for the first match
\           Repeat the last search
```

Like `apt-get`, `aptitude` installs packages upon which a selected package Depends. `aptitude` also offers the option to pull in packages that a to-be-installed package Recommends or Suggests. You can change the default behavior by choosing `F10 -> Options -> Dependency handling` in its menu.

Other advantages of `aptitude` are:

- `aptitude` offers access to all versions of a package.
- `aptitude` logs its actions in `/var/log/aptitude`.
- `aptitude` makes it easy to keep track of obsolete software by listing under "Obsolete and Locally Created Packages".
- `aptitude` includes a fairly powerful system for searching particular packages and limiting the package display. Users familiar with `mutt` will pick up quickly, as mutt was the inspiration for the expression syntax. See "SEARCHING, LIMITING, AND EXPRESSIONS" in `/usr/share/doc/aptitude/README`.
- `aptitude` in full screen mode has `su` functionality embedded and can be run from normal user until you really need administrative privileges.

3.2.4 `dselect`

In stable releases up to and including Potato, `dselect` was the principal package maintenance tool. For Sarge, you should consider using `aptitude` instead.

When started, `dselect` automatically selects all "Required", "Important", and "Standard" packages.

`dselect` has a somewhat strange user interface. Most people get used to it, however. It has four commands (Capital means CAPITAL!):

```
Key-stroke  Action
Q           Quit. Confirm current selection and quit anyway.
            (override dependencies)
R           Revert! I did not mean it.
D           Damn it! I do not care what dselect thinks.  Just Do it!
U           Set all to sUggested state
```

With `D` and `Q`, you can select conflicting selections at your own risk. Handle these commands with care.

Add a line containing the option "expert" in `/etc/dpkg/dselect.cfg` to reduce noise.

If your machine runs `dselect` slowly then you might consider running `dselect` on another (faster) machine in order to determine the packages you want to install, then use `apt-get install` on the slow machine to install them.

3.2.5 Tracking a distribution using APT

To track the `testing` distribution as it changes, make your `/etc/apt/preferences` file look like this:

```
Package: *
Pin: release a=testing
Pin-Priority: 800

Package: *
Pin: release a=stable
Pin-Priority: 600
```

Note that tracking the `testing` distribution can have the side effect of delaying the installation of packages containing security fixes. Such packages are uploaded to `unstable` and migrate to `testing` only after a delay.

See `apt_preferences(5)` for more complicated examples which will allow you, for example, to track `testing` while installing selected packages from `unstable`.

Examples which lock particular packages at particular versions while tracking other packages as they are released are available in the examples subdirectory (`http://www.debian.org/doc/manuals/debian-reference/examples/`) as `preferences.testing` and `preferences.unstable`.

If you mix distributions, e.g., `testing` with `stable` or `unstable` with `stable`, you will eventually pull in core packages such as `libc6` from `testing` or `unstable` and there is no guarantee that these will not contain bugs. You have been warned.

Another example, `preferences.stable`, forces all packages to be downgraded to `stable`.

Downgrading from a later release of a **package** to an earlier one is not officially supported in Debian. However, you may find that you have to downgrade a specific package in order to re-install a version of a package that works when a new version malfunctions. You may find these previous package files locally in `/var/cache/apt/archives/` or remotely at `http://snapshot.debian.net/`. See also 'Rescue using `dpkg`' on page 12.

Downgrading from a later release of a **distribution** to an earlier one is not officially supported either and is very likely to cause problems. However, this may be worth trying as a last resort if you are desperate.

3.2.6 `aptitude`, `apt-get` and `apt-cache` commands

While tracking `testing` as described in the above example you can manage the system by using the following commands:

- `aptitude update` (or `apt-get update`)

  These update the list of available packages at the repositories.

- `aptitude upgrade` (or `apt-get upgrade` or `aptitude dist-upgrade` or `apt-get dist-upgrade`)

These track the `testing` distribution — they upgrade each package on the system, after installing versions of packages upon which it Depends, from the `testing` distribution. [2]

- `apt-get dselect-upgrade`

  This tracks the `testing` distribution — it upgrades each package on the system according to the selections of `dselect`.

- `aptitude install` *`package`*`/unstable`

  This installs *package* from the `unstable` distribution while installing its dependencies from the `testing` distribution.

- `aptitude install -t unstable` *`package`*

  This installs *package* from the `unstable` distribution while installing its dependencies also from the `unstable` distribution by setting the Pin-Priority of `unstable` to 990.

- `apt-cache policy` *`foo bar ...`*

  This checks the status of packages *foo bar ...*.

- `aptitude show` *`foo bar ...`* `| less` (or `apt-cache show` *`foo bar ...`* `| less`)

  This checks the information for packages *foo bar ...*.

- `aptitude install` *`foo=2.2.4-1`*

  This installs the particular version *2.2.4-1* of the *foo* package.

- `aptitude install` *`foo bar-`*

  This installs the *foo* package and removes the *bar* package

- `aptitude remove` *`bar`*

  This removes the *bar* package but not its configuration files.

- `aptitude purge` *`bar`*

  This removes the *bar* package together with all its configuration files.

In the above examples, giving `apt-get` the `-u` option causes it to print a list of all packages that are to be upgraded and to prompt the user before taking action. `aptitude` does this by default. The following makes `apt-get` always do this:

```
$ cat >> /etc/apt/apt.conf << .
// Always show packages to be upgraded (-u)
APT::Get::Show-Upgraded "true";
.
```

[2] The difference between `upgrade` and `dist-upgrade` only appears when new versions of packages stand in different dependency relationships from old versions of those packages. See `apt-get(8)` for details. `aptitude upgrade` and `aptitude dist-upgrade` start `aptitude` in the commandline mode. You can switch these to full screen mode by pressing `e` key.

Use the `--no-act` option to simulate actions without actually installing, removing, etc., any packages.

3.3 Debian survival commands

With this knowledge you can live the life of eternal upgrade :-)

3.3.1 Check bugs in Debian and seek help

If you are experiencing problems with a specific package, make sure to check out these sites first before you seek help or file a bug report. (`lynx`, `links`, and `w3m` work equally well):

```
$ lynx http://bugs.debian.org/
$ lynx http://bugs.debian.org/package-name  # if you know package name
$ lynx http://bugs.debian.org/bugnumber     # if you know bug number
```

Search Google (www.google.com) with search words including "site:debian.org".

When in doubt, read the fine manual. Set `CDPATH` as follows:

```
export CDPATH=.:/usr/local:/usr/share/doc
```

and type

```
$ cd packagename
$ pager README.Debian # if this exists
$ mc
```

3.3.2 APT upgrade troubleshooting

Package dependency problems may occur when upgrading in `unstable` or `testing` as described in 'Upgrading' on page 4. Most of the time this is because a package that will be upgraded Depends on a package that is not yet available. These problems are fixed by using

```
# aptitude dist-upgrade
```

If this does not work, then repeat one of the following until the problem resolves itself:

```
# aptitude -f upgrade        # continue upgrade even after error
... or
# aptitude -f dist-upgrade   # continue dist-upgrade even after error
```

Some really broken upgrade scripts may cause persistent trouble. It is usually better to resolve this type of situation by inspecting the `/var/lib/dpkg/info/packagename.{post,pre}{inst,rm}` scripts of the offending package and then running:

```
# dpkg --configure -a    # configures all partially installed packages
```

If a script complains about a missing configuration file, look in `/etc/` for the corresponding configuration file. If one exists with an extension of `.dpkg-new` (or something similar), `mv` it to remove the suffix.

Package dependency problems may occur when installing in `unstable` or `testing`. There are ways to circumvent dependencies.

```
# aptitude -f install package # override broken dependencies
```

An alternative method to fix these situations is to use the `equivs` package. See `/usr/share/doc/equivs/README.Debian`.

3.3.3 Rescue using dpkg

If you reach a dead end using APT you can download package files from Debian mirrors and install them using `dpkg`. If you do have not access to the network you can look for cached copies of package files in `/var/cache/apt/archives/`.

```
# dpkg -i fetchmail_6.2.5-4_i386.deb
```

If attempting to install a package this way fails due to dependency violations and you really need to install the package then you can override dependency checks using `dpkg`'s `--ignore-depends`, `--force-depends` and other options. See `dpkg(8)` for details.

3.3.4 Recover package selection data

If `/var/lib/dpkg/status` becomes corrupt for any reason, the Debian system loses package selection data and suffers severely. Look for the old `/var/lib/dpkg/status` file at `/var/lib/dpkg/status-old` or `/var/backups/dpkg.status.*`.

Keeping `/var/backups/` in a separate partition may be a good idea since this directory contains lots of important system data.

If no old `/var/lib/dpkg/status` file is available, you can still recover information from directories in `/usr/share/doc/`.

```
# ls /usr/share/doc | \
  grep -v [A-Z] | \
  grep -v '^texmf$' | \
  grep -v '^debian$' | \
  awk '{print $1 " install"}' | \
  dpkg --set-selections
# dselect --expert # reinstall system, de-select as needed
```

3.3.5 Rescue system after crashing `/var`

Since the `/var` directory contains regularly updated data such as mail, it is more susceptible of corruption than, e.g., `/usr/`. Putting `/var/` on a separate partition reduces risks. If disaster happens, you may have to rebuild the `/var` directory to rescue your Debian system.

Obtain the skeleton content of the `/var` directory from a minimum working Debian system based on the same or older Debian version, for example `var.tar.gz(http://people.debian.org/~osamu/pub/)`, and place it in the root directory of the broken system. Then

```
# cd /
# mv var var-old      # if any useful contents are left
# tar xvzf var.tar.gz # use Woody skeleton file
# aptitude            # or dselect
```

This should provide a working system. You can expedite the recovery of package selections by using the technique described in 'Recover package selection data' on the facing page. ([FIXME]: This procedure needs more experiments to verify.)

3.3.6 Install a package into an unbootable system

Boot into Linux using a Debian rescue floppy/CD or an alternative partition in a multiboot Linux system. Mount the unbootable system on `/target` and use the chroot install mode of `dpkg`.

```
# dpkg --root /target -i packagefile.deb
```

Then configure and fix problems.

By the way, if a broken `lilo` is all that prevents booting, you can boot using a standard Debian rescue disk. At boot prompt, assuming the root partition of your Linux installation is in `/dev/hda12` and you want runlevel 3, enter:

```
boot: rescue root=/dev/hda12 3
```

Then you are booted into an almost fully functional system with the kernel on floppy disk. (There may be minor glitches due to lack of kernel features or modules.)

3.3.7 What to do if the `dpkg` command is broken

A broken `dpkg` may make it impossible to install any `.deb` files. A procedure like the following will help you recover from this situation. (In the first line, you can replace "links" with your favorite browser command.)

```
$ links http://http.us.debian.org/debian/pool/main/d/dpkg/
  ... download the good dpkg_version_arch.deb
$ su
password: *****
# ar x dpkg_version_arch.deb
# mv data.tar.gz /data.tar.gz
# cd /
# tar xzfv data.tar.gz
```

For `i386`, `http://packages.debian.org/dpkg` may also be used as the URL.

3.4 Debian nirvana commands

Enlightenment with these commands will save a person from the eternal karmic struggle of upgrade hell and let him reach Debian **nirvana**. :-)

3.4.1 Information on a file

To find the package to which a particular filename pattern belongs in the installed packages:

```
$ dpkg {-S|--search} pattern
```

Or to find the similar in the Debian archive:

```
$ wget http://ftp.us.debian.org/debian/dists/sarge/Contents-i386.gz
$ zgrep -e pattern Contents-i386.gz
```

Or use specialized package commands:

```
# aptitude install dlocate
$ dlocate filename          # fast alternative to dpkg -L and dpkg -S
...
# aptitude install auto-apt # on-demand package installation tool
# auto-apt update           # create db file for auto-apt
$ auto-apt search pattern
                # search for pattern in all packages, installed or not
```

3.4.2 Information on a package

Search and display information from package archives. Make sure to point APT to the proper archive(s) by editing `/etc/apt/sources.list`. If you want to see how packages in `testing/unstable` do against the currently installed one, use `apt-cache policy`—quite nice.

```
# apt-get   check           # update cache and check for broken packages
$ apt-cache search  pattern # search package from text description
$ apt-cache policy  package # package priority/dists information
$ apt-cache show -a package # show description of package in all dists
$ apt-cache showsrc package # show description of matching source package
$ apt-cache showpkg package # package information for debugging
# dpkg  --audit|-C          # search for partially installed packages
$ dpkg {-s|--status} package ... # description of installed package
$ dpkg -l package ...       # status of installed package (1 line each)
$ dpkg -L package ...       # list filenames installed by the package
```

`apt-cache showsrc` is not documented as of the Woody release but works :)

You can also find package information in (I use `mc` to browse these):

```
/var/lib/apt/lists/*
/var/lib/dpkg/available
```

The comparison of the following files provides information on what exactly has happened in the last few install sessions.

```
/var/lib/dpkg/status
/var/backups/dpkg.status*
```

3.4.3 Unattended installation with APT

For an unattended installation, add the following line in `/etc/apt/apt.conf`:

```
Dpkg::Options {"--force-confold";}
```

This equivalent to running `aptitude -y install packagename` or `apt-get -q -y install packagename`. Because this automatically answers "yes" to all prompts, it may cause problems, so use this trick with care. See `apt.conf(5)` and `dpkg(1)`.

You can configure any particular packages later by following 'Reconfigure installed packages' on the next page.

3.4.4 Reconfigure installed packages

Use the following to reconfigure any already-installed package.

```
# dpkg-reconfigure --priority=medium package [...]
# dpkg-reconfigure --all   # reconfigure all packages
# dpkg-reconfigure locales # generate any extra locales
# dpkg-reconfigure --p=low xserver-xfree86 # reconfigure X server
```

Do this for `debconf` if you need to change the `debconf` dialog mode permanently.

Some programs come with special configuration scripts. [3]

```
apt-setup     - create /etc/apt/sources.list
install-mbr   - install a Master Boot Record manager
tzconfig      - set the local time zone
gpmconfig     - set gpm mouse daemon
eximconfig    - configure Exim (MTA)
texconfig     - configure teTeX
apacheconfig  - configure Apache (httpd)
cvsconfig     - configure CVS
sndconfig     - configure sound system
...
update-alternatives - set default command, e.g., vim as vi
update-rc.d         - System-V init script management
update-menus        - Debian menu system
...
```

3.4.5 Remove and purge packages

Remove a package while maintaining its configuration:

```
# aptitude remove package ...
# dpkg   --remove package ...
```

Remove a package and all configuration:

```
# aptitude purge  package ...
# dpkg   --purge  package ...
```

[3] Some `*config` scripts are disappearing in the newer Sarge release and the package configuration functionality is moved to the `debconf` system.

3.4.6 Holding older packages

For example, holding of `libc6` and `libc6-dev` for `dselect` and `aptitude install` *`package`* can be done as follows:

```
# echo -e "libc6 hold\nlibc6-dev hold" | dpkg --set-selections
```

`aptitude install` *`package`* will not be hindered by this "hold". To hold a package through forcing automatic downgrade for `aptitude upgrade` *`package`* or `aptitude dist-upgrade`, add the following to `/etc/apt/preferences`:

```
Package: libc6
Pin: release a=stable
Pin-Priority: 2000
```

Here the "`Package:`" entry cannot use entries such as "`libc6*`". If you need to keep all binary packages related to the `glibc` source package in a synchronized version, you need to list them explicitly.

The following will list packages on hold:

```
dpkg --get-selections "*"|grep -e "hold$"
```

3.4.7 Mixed `stable/testing/unstable` system

`apt-show-versions` can list available package versions by distribution.

```
$ apt-show-versions | fgrep /testing | wc
... how many packages you have from testing
$ apt-show-versions -u
... list of upgradeable packages
$ aptitude install `apt-show-versions -u -b | fgrep /unstable`
... upgrade all unstable packages to their newest versions
```

3.4.8 Prune cached package files

Package installation with APT leaves cached package files in `/var/cache/apt/archives/` and these need to be cleaned.

```
# aptitude autoclean # removes only useless package files
# aptitude clean     # removes all cached package files
```

3.4.9 Record/copy system configuration

To make a local copy of the package selection states:

```
# dpkg --get-selections "*" >myselections   # or use \*
# debconf-get-selections > debconfsel.txt
```

"*" makes *myselections* include package entries for "purge" too.

You can transfer this file to another computer, and install it there with:

```
# dselect update
# debconf-set-selections < debconfsel.txt
# dpkg --set-selections <myselections
# apt-get -u dselect-upgrade    # or dselect install
```

3.4.10 Port a package to the `stable` system

For partial upgrades of the `stable` system, rebuilding a package within its environment using the source package is desirable. This avoids massive package upgrades due to their dependencies. First, add the following entries to `/etc/apt/sources.list`:

```
deb-src http://http.us.debian.org/debian testing \
 main contrib non-free
deb-src http://http.us.debian.org/debian unstable \
 main contrib non-free
```

Here each entry for `deb-src` is broken into two lines because of printing constraints, but the actual entry in `sources.list` should consist of a single line.

Then get the source and make a local package:

```
$ apt-get update  # update the source package search list
$ apt-get source package
$ dpkg-source -x package.dsc
$ cd package-version
  ... inspect required packages (Build-Depends in .dsc file) and
      install them too.  You need the "fakeroot" package also.

$ dpkg-buildpackage -rfakeroot

  ...or (no sig)
$ dpkg-buildpackage -rfakeroot -us -uc # use "debsign" later if needed

  ...Then to install
$ su -c "dpkg -i packagefile.deb"
```

Usually, one needs to install a few packages with the "-dev" suffix to satisfy package dependencies. `debsign` is in the `devscripts` package. `auto-apt` may ease satisfying these dependencies. Use of `fakeroot` avoids unnecessary use of the root account.

In Woody, these dependency issues can be simplified. For example, to compile a source-only `pine` package:

```
# apt-get build-dep pine
# apt-get source -b pine
```

3.4.11 Local package archive

In order to create a local package archive which is compatible with APT and the `dselect` system, `Packages` needs to be created and package files need to be populated in a particular directory tree.

A local `deb` repository similar to an official Debian archive can be made in this way:

```
# aptitude install dpkg-dev
# cd /usr/local
# install -d pool # physical packages are located here
# install -d dists/unstable/main/binary-i386
# ls -1 pool | sed 's/_.*$/ priority section/' | uniq > override
# editor override # adjust priority and section
# dpkg-scanpackages pool override /usr/local/ \
   > dists/unstable/main/binary-i386/Packages
# cat > dists/unstable/main/Release << EOF
Archive: unstable
Version: 3.0
Component: main
Origin: Local
Label: Local
Architecture: i386
EOF
# echo "deb file:/usr/local unstable main" \
   >> /etc/apt/sources.list
```

Alternatively, a quick-and-dirty local `deb` repository can be made:

```
# aptitude install dpkg-dev
# mkdir /usr/local/debian
# mv /some/where/package.deb /usr/local/debian
# dpkg-scanpackages /usr/local/debian /dev/null | \
  gzip - > /usr/local/debian/Packages.gz
#  echo "deb file:/usr/local/debian ./" >> /etc/apt/sources.list
```

These archives can be remotely accessed by providing access to these directories through either HTTP or FTP methods and changing entries in `/etc/apt/sources.list` accordingly.

3.4.12 Convert or install an alien binary package

`alien` enables the conversion of binary packages provided in Red Hat `rpm`, Stampede `slp`, Slackware `tgz`, and Solaris `pkg` file formats into a Debian `deb` package. If you want to use a package from another Linux distribution than the one you have installed on your system, you can use `alien` to convert it to your preferred package format and install it. `alien` also supports LSB packages.

3.4.13 Automatically install command

`auto-apt` is an on-demand package installation tool.

```
$ sudo auto-apt update
 ... update database
$ auto-apt -x -y run
Entering auto-apt mode: /bin/bash
Exit the command to leave auto-apt mode.
$ less /usr/share/doc/med-bio/copyright # access non-existing file
 ...  Install the package which provide this file.
 ... Also install dependencies
```

3.4.14 Verify installed package files

`debsums` enables verification of installed package files against MD5 checksums. Some packages do not have available MD5 checksums. A possible temporary fix for sysadmins:

```
# cat >>/etc/apt/apt.conf.d/90debsums
DPkg::Post-Install-Pkgs {"xargs /usr/bin/debsums -sg";};
^D
```

per Joerg Wendland <`joergland@debian.org`> (untested).

3.5 Other Debian peculiarities

3.5.1 The `dpkg-divert` command

File **diversions** are a way of forcing `dpkg` not to install a file into its default location, but to a **diverted** location. Diversions can be used through the Debian package scripts to move a file

away when it causes a conflict. System administrators can also use a diversion to override a package's configuration file, or whenever some files (which aren't marked as conffiles) need to be preserved by `dpkg`, when installing a newer version of a package which contains those files.

```
# dpkg-divert [--add]  filename # add "diversion"
# dpkg-divert --remove filename # remove "diversion"
```

It's usually a good idea not to use `dpkg-divert` unless it is absolutely necessary.

3.5.2 The `equivs` package

If you compile a program from source, it is best to make it into a real local debianized package (`*.deb`). Use `equivs` as a last resort.

```
Package: equivs
Priority: extra
Section: admin
Description: Circumventing Debian package dependencies
 This is a dummy package which can be used to create Debian
 packages, which only contain dependency information.
```

3.5.3 Alternative commands

To make the command `vi` run `vim`, use `update-alternatives`:

```
# update-alternatives --display vi
...
# update-alternatives --config vi
  Selection    Command
-----------------------------------------------
      1        /usr/bin/elvis-tiny
      2        /usr/bin/vim
*+    3        /usr/bin/nvi

Enter to keep the default[*], or type selection number: 2
```

Items in the Debian alternatives system are kept in `/etc/alternatives/` as symlinks.

To set your favorite X Window environment, apply `update-alternatives` to `/usr/bin/x-session-manager` and `/usr/bin/x-window-manager`.

`/bin/sh` is a direct symlink to `/bin/bash` or `/bin/dash`. It's safer to use `/bin/bash` to be compatible with old Bashism-contaminated scripts but better discipline to use `/bin/dash` to enforce POSIX compliance. Upgrading to a 2.4 Linux kernel tends to set this to `/bin/dash`.

3.5.4 Runlevel usage

When installed, most Debian packages configure their services to run in runlevels 2 through 5. Thus, there are no differences between runlevels 2, 3, 4 and 5 on a Debian system that has not been customized; Debian leaves it up to the local administrator to customize runlevels. This differs from the way runlevels are used by some other popular GNU/Linux distributions. One change you may want to make is to disable `xdm` or `gdm` in runlevel 2 so that the X display manager is not started at the end of the boot sequence; you can then start it by switching to runlevel 3.

3.5.5 Disabled daemon services

Debian developers take system security seriously. Many daemon services are installed with the fewest services and features enabled.

Run `ps aux` or check the contents of `/etc/init.d/*` and `/etc/inetd.conf`, if you have any doubts (about Exim, DHCP, ...). Also check `/etc/hosts.deny`. The `pidof` command is also useful (see `pidof(8)`).

X11 doesn't allow TCP/IP (remote) connections by default in recent versions of Debian. X forwarding in SSH is also disabled.

Appendix A

Appendix

A.1 Authors

Debian Quick Reference was initiated by Osamu Aoki <osamu\#at\#debian.org> as a personal installation memo that was eventually called "Quick Reference ...". Many contents came from the archives of the "debian-user" mailing list. Also "Debian Installation Manual" and "Debian Release Notes" were referenced.

Following a suggestion from Josip Rodin, who is very active with the Debian Documentation Project (http://www.debian.org/doc/ddp) (DDP) and is the current maintainer of "The Debian FAQ", this document was renamed as "Debian Reference" and was merged with several chapters from the "The Debian FAQ" with reference-like contents. Then "Debian Quick Reference" was formed as an excerpt.

This document has been edited, translated, and expanded by the following QREF team members:

- English originals for original "Quick Reference..."
  - Osamu Aoki <osamu\#at\#debian.org> (leader: all contents)
- English proofreading and additional contribution
  - Esko Arajärvi <edu\#at\#iki.fi> (etch updates)
  - Thomas Hood <jdthood\#at\#yahoo.co.uk> (network related)
  - Brian Nelson <nelson\#at\#bignachos.com> (especially X related)
  - David Sewell <dsewell\#at\#virginia.edu> (retired)
  - Jan Michael C Alonzo <jmalonzo\#at\#spaceants.net>
  - Daniel Webb <webb\#at\#robust.colorado.edu>
  - Feedback from all translators
- French translation
  - Guillaume Erbs <gerbs\#at\#free.fr> (leader: fr)
  - Rénald Casagraude <rcasagraude\#at\#interfaces.fr>
  - Jean-Pierre Delange <adeimantos\#at\#free.fr>
  - Daniel Desages <daniel\#at\#desages.com>
- Italian translation
  - Davide Di Lazzaro <mc0315\#at\#mclink.it> (leader: it)

- Portuguese (Brazil) translation
  - Paulo Rogério Ormenese <pormenese\#at\#uol.com.br> (leader: pt-br)
  - Andre Luis Lopes <andrelop\#at\#ig.com.br>
  - Marcio Roberto Teixeira <marciotex\#at\#pop.com.br>
  - Rildo Taveira de Oliveira <to_rei\#at\#yahoo.com>
  - Raphael Bittencourt Simoes Costa <raphael-bsc\#at\#bol.com.br>
  - Gustavo Noronha Silva <kov\#at\#debian.org> (coordinator)
- Spanish translation
  - Walter Echarri <wecharri\#at\#infovia.com.ar> (leader: es)
  - José Carreiro <ffx\#at\#urbanet.ch>
- German translation
  - Jens Seidel <tux-master\#at\#web.de> (leader: de)
  - Willi Dyck <wdyck\#at\#gmx.net>
  - Stefan Schröder <stefan\#at\#fkp.uni-hannover.de>
  - Agon S. Buchholz <asb\#at\#kefk.net>
- Polish translation—the following members of PDDP (http://debian.linux.org.pl):
  - Marcin Andruszkiewicz
  - Mariusz Centka <mariusz.centka\#at\#debian.linux.org.pl>
  - Bartosz Feński <fenio\#at\#debian.linux.org.pl> (leader: pl)
  - Radosław Grzanka <radekg\#at\#debian.linux.org.pl>
  - Bartosz 'Xebord' Janowski
  - Jacek Lachowicz
  - Rafał Michaluk
  - Leonard Milcin, Jr.
  - Tomasz Z. Napierała <zen\#at\#debian.linux.org.pl>
  - Oskar Ostafin <cx\#at\#debian.linux.org.pl>
  - Tomasz Piękoś
  - Jacek Politowski
  - Mateusz Prichacz <mateusz\#at\#debian.linux.org.pl>
  - Marcin Rogowski
  - Paweł Różański
  - Mariusz Strzelecki
  - Krzysztof Ścierski
  - Przemysław Adam Śmiejek <tristan\#at\#debian.linux.org.pl>
  - Krzysztof Szynter
  - Mateusz Tryka <uszek\#at\#debian.linux.org.pl>
  - Cezary Uchto
  - Krzysztof Witkowski <tjup\#at\#debian.linux.org.pl>
  - Bartosz Zapałowski <zapal\#at\#debian.linux.org.pl>
- Chinese (simplified) translation
  - Hao "Lyoo" LIU <iamlyoo\#at\#163.net>
  - Ming Hua <minghua\#at\#rice.edu>
  - Xiao Sheng Wen <atzlinux\#at\#163.com> (leader: zh-cn)
  - Haifeng Chen <optical.dlz\#at\#gmail.com>
  - Xie Yanbo <xieyanbo\#at\#gmail.com>

  - easthero <easthero\#at\#gmail.com>
- Chinese (traditional) translation
  - Asho Yeh <asho\#at\#debian.org.tw> (leader: zh-tw)
  - Tang Wei Ching <wctang\#at\#csie.nctu.edu.tw> (ex-leader: zh-tw)
- Japanese translation
  - Shinichi Tsunoda <tsuno\#at\#ngy.1st.ne.jp> (leader: ja)
  - Osamu Aoki <osamu\#at\#debian.org>
- Finnish translation
  - Esko Arajärvi <edu\#at\#iki.fi> (leader: fi)

A.2 Warranties

Since I am not an expert, I do not pretend to be fully knowledgeable about Debian or Linux in general. Security considerations I use may only be applicable for home use.

This document does not replace any authoritative guides.

All warranties are disclaimed. All trademarks are property of their respective trademark owners.

A.3 Feedback

Comments and additions to this document are always welcome. Please send email to the Debian BTS system (http://bugs.debian.org/) under the debian-reference package or under the respective translation packages. Use of reportbug makes it easy to file a thorough bug report. You may still send email to Osamu Aoki (http://people.debian.org/~osamu/) at <osamu\#at\#debian.org> in English or to each translator in their respective language.