Debian Quick Reference

Osamu Aoki <debian@aokiconsulting.com>
Editor: David Sewell <dsewell@virginia.edu>
'Authors' on page 21

CVS, Sun, 13 Oct 2002 22:40:23 -0600

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

Keep it short and simple (KISS) is my guiding principle.

1.1 Document conventions

This "Debian Quick Reference" document provides information through short Bash shell commands.

Reference to:
- a **Unix 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 comes in 3 release "flavors":

- stable: Good to track on a production server. Boring for the workstation (WS).
- testing: Good to track on the WS.
- unstable: Never track this blindly.

Read at least the key mailing list `debian-devel-announce@lists.debian.org` for updates on the status of Debian.

In March 2002, these three release versions corresponded to `potato` (production quality), `woody` (beta-test, very stable then since release was imminent), and `sid` (alpha-test). In August 2002, right after the `woody` release, these corresponded to `woody` (production quality), `sarge` (beta-test, it will be a somewhat rough ride for some time), and `sid` (always alpha-test). 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`.

In theory, there are two things you can do get the latest versions of software running.

- 'Install system with APT' on page 6 (mainly for WS purposes)
- 'Port a package to the `stable` system' on page 15 (mainly for server purposes)

Chapter 2

Upgrading a distribution

2.1 Upgrade to “testing”

After the above preparation, the system can be upgraded.

2.1.1 Best upgrade practice 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 packages in "suggests" and "recommends"
# dselect install
```

`dselect` always works :) If you need to upgrade without `dselect` after Woody, consider `aptitudes` and other options.

2.2 Woody configuration

For a freshly installed Woody system, edit `/etc/apt/sources.list`, `/etc/apt/apt.conf`, and `/etc/apt/preferences` to achieve the same structure as described in the above section.

APT in Potato did not have the functions described in `apt_preferences(5)`.

Chapter 3

Debian package management

Make sure to set up a local HTTP proxy using `squid` for packages downloaded by APT. This greatly improves the performance of network upgrades, especially with multiple Debian boxes on the LAN. This chapter is based on a Woody system but most information also applies to a Potato system (except for `apt_preferences(5)` and topics related to `/etc/preferences`).

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 tools

```
dselect   -- menu-driven package management tool (top level)
dpkg      -- install package (package-file centric)
apt-get   -- install package (package-archive centric, CLI APT)
tasksel   -- install task (a set of packages)
aptitude  -- install package (package & task, ncurses APT)
deity     -- alternative ncurses APT
synaptic, gsynaptic -- GUI APT alternatives
```

These are not equal-level tools. `dselect` runs on the top of APT (the command-line command is `apt-get`) and `dpkg`.

APT uses `/var/lib/apt/lists/*` for tracking package status while `dpkg` uses `/var/lib/dpkg/status`. If you have installed packages directly using `apt-get` or similar programs such as aptitude, make sure to update the /var/lib/dpkg/status file from the [U]pdate selection menu in dselect or from the shell command line "dselect update" prior to running dselect select, tasksel or dpkg -l.

As for package dependencies, apt-get automatically pulls in packages with **depends** but leaves packages with **recommends** and **suggests**, while dselect offers fine-grained control over choices of these packages and pulls in **depends** and **recommends** by default.

3.1.2 Convenience tools

```
apt-cache         - check package archive in local cache
dpkg-reconfigure  - reconfigure an already installed package
dpkg-source       - manage source package file
dpkg-buildpackage - automate the building of a package file
...
```

3.2 Debian survival commands

With this knowledge, one can live a life of **eternal** "upgrade" :-)

3.2.1 Install *task* with tasksel

tasksel is the **Debian Task Installer**, which is offered as the "simple" option during system installation.

When one needs to install a common function which requires multiple packages, this is the best way to do it. Make sure to run the commands as follows:

```
# dselect update
# tasksel
```

3.2.2 Install system with APT

You can selectively install packages from different archives using newer versions of apt-get (>Woody). This enables, for example, selective upgrade to unstable and selective downgrade to stable while tracking testing.

For selective upgrade, add the sources for unstable (testing if you use stable) to your /etc/apt/sources.list and edit /etc/apt/preferences as follows:

```
Package: *
Pin: release a=unstable
Pin-Priority: 50
```

(replace `unstable` with `testing` if you normally use `stable`).

Now you can run `apt-get install` *`package`*`/unstable` and install a package out of `unstable`, with all its depends. But normal `apt-get upgrade` or `apt-get install` *`package`* does not install a package from `unstable` (or `testing`).

```
# apt-cache policy libc6 libc6-dev locales         # check status
# apt-get install libc6=2.2.4-1 libc6-dev=2.2.4-1 locales=2.2.4-1
# apt-get install libc6/unstable libc6-dev/unstable locales/unstable
# apt-get install -t unstable libc6 libc6-dev locales
# apt-get -u install interesting-new-package remove-package-
# apt-get remove useless-old-package
# apt-get remove --purge really-useless-old-package
```

To downgrade all packages to `stable`, edit `/etc/apt/preferences` as follows:

```
Package: *
Pin: release a=stable
Pin-Priority: 1001
```

and run "`apt-get upgrade`", which forces downgrade due to Pin-priority > 1000.

3.2.3 Upgrade system with APT

Upgrade system with APT:

```
# apt-get update
... then do one of the following:
# apt-get -u upgrade         # pull all depends
# apt-get -u dist-upgrade    # pull all depends and resolve dependency
# apt-get -u dselect-upgrade # follow selections set by dselect
```

The following sets the `-u` option as the default action:

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

Use the `-s` option to simulate upgrade without actual upgrade.

`dselect` offers a menu-driven interface over APT. `deity` and `aptitude` will offer alternatives to `dselect`.

3.2.4 Check bugs in Debian and seek help

If you are experiencing problems regarding a specific package, make sure to check out these sites first before you seek help or before you 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
$ mc
```

3.2.5 APT upgrade troubleshooting

Package dependency problems may occur when upgrading in `unstable/testing`. Most of the time, this is because a package that will be upgraded has a new dependency that isn't met. These problems are fixed by using

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

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

```
# apt-get upgrade -f         # continue upgrade even after error
... or
# apt-get dist-upgrade -f    # 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-}{install,remova`* 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 `.new` (or something similar), change (`mv`) it to remove the suffix.

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

```
# apt-get install -f package # override broken dependencies
```

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

3.2.6 Rescue using dpkg

Ad hoc recovery of a crashed `dselect` (APT) can be done on a really broken system by just using `dpkg` without APT:

```
# cd /var/cache/apt/archives
# dpkg -i libc6* libdb2* perl*
# dpkg -i apt* dpkg* debconf*
# dpkg -i *  # until no error occurs
```

If a package is missing, get it from mirror sites (http://www.debian.org/misc/README.mirrors) by:

```
# mc            # use "FTP link" pointing to Debian FTP server
```

As of recently, actual packages on the HTTP/FTP server may not be located under the classic `/dist` directory but rather under the new `/pool` directory.

Then install by:

```
# dpkg -i /var/cache/apt/archives/packagefile.deb
```

For a broken dependency, fix it or use:

```
# dpkg --ignore-depends=package1,... -i packagefile.deb
# dpkg --force-depends -i packagefile.deb
# dpkg --force-depends --purge package
# dpkg --force-confmiss -i packagefile.deb # Install missing conffile
```

3.2.7 Rescue system after erasing `/var`

If everything under the `/var` directory is erased, you can recover control of the system if you have backup copies of `/var/lib/dpkg/status` by:

```
# cd /
# install -d /var/cache/apt/archives
# install -d /var/cache/apt/archives/partial
# install -d /var/lib/dpkg/
# cp status-old /var/lib/dpkg/status
# apt-cache gencaches
```

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.

3.2.8 Install a package into an unbootable system

Boot into Linux using a Debian rescue floppy/CD or an alternative partition in a multi-boot 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.)

3.2.9 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
$ ar x dpkg_version_arch.deb
$ su
password: *****
# 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.

3.3 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**. :-)

3.3.1 Information on a file

To find the package to which a particular file belongs:

```
$ dpkg {-S|--search} pattern # search for pattern in installed packages
$ zgrep -e pattern /local/copy/of/debian/woody/Contents-i386.gz
             # find filename-pattern of files in the debian archive
```

Or use specialized package commands:

```
# apt-get install dlocate
                 # conflicts with slocate (secure version of locate)
$ dlocate filename         # fast alternative to dpkg -L and dpkg -S
...
# apt-get 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
```

3.3.2 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 file names 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|status}
```

3.3.3 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.

```
apt-setup     - create /etc/sources.list
install-mbr   - install a Master Boot Record manager
tzconfig      - set the local timezone
```

```
gpmconfig     - set gpm mouse daemon
sambaconfig   - configure Samba in Potato (Woody uses debconf)
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
...
```

3.3.4 Remove and purge packages

Remove a package while maintaining its configuration:

```
# apt-get remove package ...
# dpkg  --remove package ...
```

Remove a package and all configuration:

```
# apt-get remove --purge package ...
# dpkg    --purge        package ...
```

3.3.5 Holding older packages

For example, holding of `libc6` and `libc6-dev` for `dselect` and `apt-get -u upgrade` *package* can be done as follows:

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

`apt-get -u install` *package* will not be hindered by this "hold". To hold a package through forcing automatic downgrade for `apt-get -u upgrade` *package* or `apt-get -u 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$"
```

3.3.6 dselect – global configuration

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

When started, `dselect` automatically selects all "Required", "Important", and "Standard" packages. In the Potato system, some large programs such as teTeX and Emacs used to belong here and were best skipped for the initial install by manually unselecting them (by typing '_'). In Woody, these have moved to the "Optional" package category.

`dselect` has a somewhat strange user interface. There are 4 ambiguous 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. For a slower machine, run `dselect` on another fast machine to find packages and use `apt-get install` to install them. `apt-get dselect-upgrade` best honors `dselect` selections.

3.3.7 Prune cached package files

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

```
# apt-get autoclean # removes only useless package files
# apt-get clean     # removes all cached package files
```

3.3.8 Record/copy system configuration

To make a local copy of the package selection states:

```
$ dpkg --get-selections "*" >myselections   # or use \*
```

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

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

```
# apt-get update
# dpkg --set-selections <myselections
# apt-get -u dselect-upgrade
```

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

Here each entry for `deb-src` is broken into 2 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 source package/unstable
$ 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

  ...or (no sig)
$ fakeroot ./debian/rules binary
$ fakeroot ./debian/rules clean
$ cd ..
$ fakeroot dpkg-source -b package-version
  ...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
```

3.3.10 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. Install `dpkg-dev` package and:

```
# cd /usr/local
# install -d pool # physical packages are located here
# install -d dists/unstable/main/binary-i386
# ls -1 pool | sed 's/_.*$/ extra BOGUS/' | uniq > override
# editor override # adjust BOGUS
# 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
```

3.3.11 Convert or install an alien binary package

`alien` enables the conversion of binary packages provided in Redhat `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.

3.3.12 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).

3.4 Other Debian peculiarities

3.4.1 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` when it is not absolutely necessary.

3.4.2 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.
```

3.4.3 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 manager, use `x-window-manager` instead.

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

3.4.4 System-V **init** and runlevels

The default runlevel to boot into can be set in `/etc/inittab`.

Unlike other distributions, Debian makes the management of runlevel completely the sysadmin's responsibility. Management of System-V style `init` on Debian is intended to be performed through `update-rc.d` scripts.

Starting `/etc/init.d/`*`name`* in runlevel 1,2,3 and stopping in 4,5 with sequencing priority number 20 (normal) can be done by:

```
# update-rc.d name start 20 1 2 3 . stop 20 4 5 .
```

Removing symbolic links while the script in `init.d` still exists can be done by:

```
# update-rc.d -f name remove
```

For editing runlevels, I cheat. I edit entries manually using the `mv` command at the shell prompt of `mc` while copying link entries using `Alt-Enter`. For example:

```
# mv S99xdm K99xdm # disable xdm (X display manager)
```

I even disable a daemon by inserting `exit 0` at the start of an `init.d` script as a quick hack. These are `conffiles` after all.

3.4.5 Disabled daemon services

The Debian distribution takes system security seriously and expects the system administrator to be competent. Thus, sometimes ease of use appears to be a secondary concern and many daemon services come with the highest security level, with the fewest services (or none) available as their default install state.

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 <debian@aokiconsulting.com> 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 excerpts.

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

- English originals for original "Quick Reference..."
  - Osamu Aoki <debian@aokiconsulting.com> (leader: all contents)
- English proofreading and rewriting
  - David Sewell <dsewell@virginia.edu> (leader: en style)
  - Brian Nelson <nelson@bignachos.com>
  - Daniel Webb <webb@robust.colorado.edu>
- French translation
  - Guillaume Erbs <gerbs@free.fr> (leader: fr)
  - Rénald Casagraude <rcasagraude@interfaces.fr>
  - Jean-Pierre Delange <delange@imaginet.fr>
  - Daniel Desages <daniel@desages.com>
- Italian translation
  - Davide Di Lazzaro <mc0315@mclink.it> (leader: it)
- Spanish translation

– Walter Echarri <wecharri@infovia.com.ar>
– José Carreiro <ffx@urbanet.ch>

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 the 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 Osamu Aoki (http://www.aokiconsulting.com/) <debian@aokiconsulting.com> in English or to each translator in their respective language.