Debian Linux
Tips and Tricks

Table of Contents

1. Basic configuration program is : `/usr/sbin/base-config`
2. Install package groups with: `tasksel`
3. Install individual packages with :
   `dselect` or `apt-get install` *`packagename`*
4. Configuration program for system: `debconf`
5. To configure postscript fonts for a postscript printer:
   ```
   defoma-psfont-installer
   ```
6. Configuration for kdm to manage remote X servers are in:
   `/etc/kde2/kdm/Xaccess` and `/etc/kde2/kdm/kdmrc`
7. Xserver DPI settings ca be changed from:
   ```
   /etc/kde2/kdm/Xservers
   ```
8. Edit the following files to allow TrueType fonts:
   `/etc/X11/XF86Config-4`
   `/etc/X11/fs/config`
   `/etc/X11/fs-xtt/config` (for xtt true-type fonts server)
   Data to enter:
   ```
   FontPath "/var/lib/deforma/x-ttcidfont-conf.d/dirs/CID"
   FontPath "/var/lib/deforma/x-ttcidfont-conf.d/dirs/TrueType"
   ```
9. Configuration of X-Server access control is in:
   ```
   /etc/X11/Xwrapper.config
   ```

10. X-Server keyboard symbols for keys layout meanings are in :
    `/etc/X11/xkb/symbols/` directory

11. To get rid of all the unused libraries from your Debian system
    If you have an easy-to-upgrade Linux system, you end up with a system that's been upgraded many times instead of backed up and reinstalled.
    To get rid of all the unused libraries from your Debian system, try the deborphan utility:
    `http://www.tribe.eu.org/deborphan/`
    (or, or course, `apt-get install deborphan`).
    It finds all the libraries that no longer have anything depending on them.
    To purge unused libraries, simply do this:
    ```
    deborphan | sudo xargs apt-get -y --purge remove
    ```

12. To search for a package name in All available packages:
    (installed and uninstalled):
    `apt-cache search` *`pathern`*
    `grep -r "^Package:.`*`pathern.*`*`" /var/lib/apt/lists/`

13. To list information about a NOT installed package:
    `dpkg --info` *`packagename`*`.deb` (Package's header info)
    `dpkg --content` *`packagename`*`.deb` (Package's file list)

14. To search for a filename in installed packages:
    `dpkg -S` *`search_pathern`* (`*` and ? are used)

15. To search for a package name through already installed packages:
    `dpkg -l "`*`pathern`*`"`

16. To list all already installed packages, their version and description:
    `dpkg -l | less`
    You can see the list of installed packages with `dpkg -l`.
    there is a small problem with this command, because the package names are sometimes truncated.
    for exampe `dpkg -l |grep libsasl` will give the output:

```
ii  libsasl-digest 1.5.24-11      DIGEST-MD5 Authentication Module for SASL
ii  libsasl-module 1.5.27-3.1wood Basic Pluggable Authentication Modules for S
ii  libsasl7       1.5.27-3.1wood Authentication abstraction library.
```

    the full name of the first package is actually `libsasl-digestmd5-des` and the name of the second package is `libsasl-modules-plain`.
    A quick way to find out the complete name is with the command:

```
> script
> dpkg -l
> exit
> less typescript or cat typescript | grep 'pattern'
```

    The command script saves all the further typed commands, including results, into a file called typescript, of which you can view its content after you terminated the script program with the command exit .
    In reality the program script stats a new bash and saves everything that comes onto the screen into the file typescript.

17. To display the files installed from an installed package:
    `dpkg -L` *`packagename`*

18. To display header information about an installed package:
    `dpkg --print-avail` *`packagename`*

19. To reconfigure a package:
    `dpkg-reconfigure` *`packagename`*

20. To fix not finished installations and configuration
    `dpkg-reconfigure -a`

21. To load the cache of internet packages install sources:
    `apt-get update`

22. To upgrade the curent version of Debian from the packages install sources:
    `apt-get upgrade`

23. To de-install a package:
    `dpkg -r packagename`
    to ignore dependencies while de-installing
    `dpkg -r --force-depends packagename`

24. Get a list of installed packages with their sizes sorted by size

```
dpkg-query -W --showformat='${Installed-Size} ${Package}\n' \
 | sort -n
```

25. To change temporarily the terminal keyboard layout:
    `loadkeys de-latin1-nodeadkeys`
    This will be taken from the file:

`/usr/share/keymaps/i386/qwertz/de-latin1-nodeadkeys.kmap.gz`

26. To change permanently the terminal keyboard layout:
    `dpkg-reconfigure console-data`

27. To force all kernel messages (`klogd`) to be sent only to syslog:
    (and not to console any more):
    Debian `klogd` normally sends all kernel messages of lower priority than 7(highest) to the current console as well as to `syslogd`.
    To prevent any messages to go to the console, start the klogd with the option `-c 1`.
    Add it to the variable `KLOGD` in the script `/etc/init.d/klogd.` eg. `KLOGD='-c 1'`

28. To install the IDS system on Debian we need the debian packages:
    `libimage-info-perl`
    `perlmagick`
    `libjpeg-progs` (uses `jpegtran` program)
    SuSE needs the rpm packages:
    `perl-Image-Info`
    `perl-PerlMagick`
    `jpeg` (uses `jpegtran` program)

29. To allow e-mailing of received faxes (hylafax) in a PDF format .
    The `uuencode` program is needed for that.
    (See `95_Fax_server.sdw` document for details)
    On Debian distribution the `uuencode` program is part of the package `sharutils.`

30. Where are Pre and Post the install scripts for packages:
    For Install:
    `/var/lib/dpkg/info/xxxx.preinst`
    `/var/lib/dpkg/info/xxxx.postint`
    For Remove:
    `/var/lib/dpkg/info/xxxx.prerm`
    `/var/lib/dpkg/info/xxxx.postrm`

31. To Install webmin on Debian 3.0 with SSL :
    - For SSL connections , it needs the following 2 packages to be installed:
    - `ssleay`
    - `libnet-ssleay-perl`

    - Download latest *xxx*`.tar.gz` Version (1.0.5) `http://www.webmin.com`
    - Uncompress it into a directory
    - In that directory Run the program: `setup.sh`
    - It should recognize the Debian Version and asks some questions.

32. To search through the bash history with the page-up and page-down key.
    Put this into your `/etc/inputrc` or `~/.inputrc` :

```
"\e[5~":        history-search-backward
"\e[6~":        history-search-forward
```

33.Exim Mail server:
Information in:
`eximdoc` package and in
`/usr/share/doc/exim/spec.txt`
Configuration:
`/usr/sbin/eximconfig`
example in : `/usr/share/doc/exim/example.conf.gz`

34.Network setup.
The network configuration files are in /etc/network where:

`interfaces`	Contains the interfaces settings (IP, netmask, etc)
`options`	Contains the general network options (forwarding, etc)

These settings are used by the programs `/sbin/ifup` and `/sbin/ifdown`

For PCMCIA Network cards the config/script files are in
`/etc/pcmcia/network.opts`
where you can enter all the configuration of the PCMCIA Network card or
if the Variables `DHCP="n"` and `IPADDR` is left empty
then the `cardmgr` network setup uses the settings in `/etc/network/interfaces`.

35.To create Runlevel links for daemons:
```
update -rc.d scriptname defaults
```

36.X-Server Symbolic links:
In Debian the symbolic links to X binary is in fact a small program which starts the X-Server. So:
`/usr/X11R6/bin/X` loads `/usr/X11R6/bin/XFree86`

37.GUI for APT and deb packages.
Good GUI for managing deb packages in Debian: **synaptic**:
```
apt-get install synaptic
synaptic                 And you're with a great DEB packages Interface
```

38.To install a Knoppix on Hard disk:
Prior to Version 3.4 (Up to ver. 3.3) the command to install live CD on the hard disk is:
`knx_hdinstall` (run as root)
From Version 3.4 the command is:(run as `knoppix` user, NOT as root)
```
IGNORE_CHECK=1 sudo knoppix-installer
```

39.Downgrading from `unstable` to `stable` after regretting an apt-get dist-upgrade.
Here are the steps that allow us to downgrade a `sarge` from unstable to stable.
- Create/Edit the file /etc/apt/preferences
  Enter:
```
Package: *
Pin: release a=stable
Pin-Priority: 1001
```
- Issue the command:
```
apt-get dist-upgrade
```
and the whole system will deactualized.

40. The command "`apt-get update`"
    creates the files
    `/var/cache/apt/srcpkgcache.bin` and
    `/var/cache/apt/pkgcache.bin`
    `/var/lib/apt/lists/security.debian.org_debian-security_dists_stable_updates_non-free_binary-i386_Packages`
    `/var/lib/apt/lists/security.debian....`
    `/var/lib/apt/lists/ftp2.de.debian.org_debian_dists_stable_main_binary-i386_Packages`
    etc.

41. The command "`apt-get upgrade`"
    download the packages in
    `/var/cache/apt/archives/partial`
    (during the download of the package)
    and finally put them in `/var/cache/apt/archives`

42. The command "`apt-get dist-upgrade`"
    is mainly thought to upgrade from a lower to a higher versions of Debian
    like from Potato to Woody, from Woody to Sarge etc.
    Beside making a "real" distribution upgrade, you can also use this command
    to upgrade your Sid version. This will clean intelligently the dependencies,
    really install the last package versions.

43. `dpkg -i --force-downgrade`
    sometimes it is necessary to go back to a previous version of package, for example
    when a new kernel doesn't work.
    change to the directory `/var/cache/apt/archives` and reinstall the previous
    package version:

    `dpkg -i --force-downgrade` *`package-name_version_architekture`*`.deb`
    `dpkg --force-help` for more infos.

44. The command "`apt-get autoclean`"
    remove the old packages from `/var/cache/apt/archives`
    (may free a lot of disk space!)

45. The command "`apt-get remove --purge` *`packagename`*"
    removes everything, also the configuration files.
    the command "`dpkg -P|--purge` *`packagename`*" removes as well everything.

46. The file `/var/lib/dpkg/availabe`
    contains the list of all available (installed) packages

47. The file `/var/lib/dpkg/status`
    contains the installation status of each package

48. The directory `/var/lib/dpkg/info`
    contains many files related to a package.

49. Possible content(files)of a package:

`*.preinst`	script that runs before installation
`*.postinst`	script that runs after installation
`*.prerm`	script that runs before the removing of the package
`*.postrm`,	script that runs after the removing of the package
`*.conffiles`	a list of the configuration files
`*.lists`	a list of all files included in the package
`*.md5sums`	md5 sums of some important files
`*.templates`	messages in several languages
`*.shlibs`	a list of used library

50.**Command `apt-file`**:
first "`apt-file update`" has to be called to get the "file lists" which will be stored in `/var/cache/apt/` as gziped files.
then with "`apt-file search <`*`string`*`>`" a file can be searched.
example: `apt-file search kernel-source-2.6`

51.**Command `apt-cache`**:
similar to `apt-file`.
example: `apt-cache search kernel-source-2.6`

52.**Command `findpkg`**:
see page `http://www.infodrom.org/Infodrom/tools/findpkg.html`

53.**Command `synaptic`**:
very nice graphic front-end for `apt-get`

54.To change the timezone of the system.
```
tzconfig
```

55.To reboot with another grub menu item only once:
`grub-reboot` *`ItemNr`*
`ItemNr` = menu item number starting with first item = 0

56.To finish an aborted packages upgdate:
```
dpkg   -configure -a
```

57.To change the timezone of the system:
```
tzconfig
```

58.Fixing the locales(language) in Etch for squirrelmail
- Make sure the squirrelmail-locales package is installed
- Run the command: `dpkg-reconfigure locales`
  and make sure that the `de_DE.UTF-8` is selected in the list.
- Edit the file:
  `/usr/share/squirrelmail/functions/i18n.php`
  and add the **bold** part in the following line:
```
$languages['de_DE']['LOCALE']  = array('de_DE.ISO8859-
                 1','de_DE.ISO-8859-1','de_DE','de_DE.UTF-8');
```

59.To force remove a broken package.
```
dpkg --force-remove-reinstreq --purge packagename
```
In case this operation fails it might be because of trying to stop a daemon via an init script that fails. In that case just add the command `exit 0` at the beginning of the init script and try the command again.

60.Installation of ClamAV, Spamassassin and Amavis in Debian Etch.
- Install the packages
- Add the user amavis to the group clamav (in /etc/group)
- Make sure that the user clamav is added to the group amavis as well
- Add the following lines in /etc/amavis/conf.d/20-debian_defaults

```
# MTA SETTINGS, UNCOMMENT AS APPROPRIATE,
# both $forward_method and $notify_method default to 'smtp:127.0.0.1:10025'
# POSTFIX, or SENDMAIL in dual-MTA setup, or EXIM V4
# (set host and port number as required; host can be specified
# as IP address or DNS name (A or CNAME, but MX is ignored)
$forward_method = 'smtp:127.0.0.1:10025'; # where to forward mail
$notify_method = $forward_method;         # where to submit notifications
# Net::Server pre-forking settings
# You may want $max_servers to match the width of your MTA pipe
# feeding amavisd, e.g. with Postfix the 'Max procs' field in the
# master.cf file, like the '2' in the:  smtp-amavis unix - - n - 2 smtp
$max_servers  =  2;     # number of pre-forked children         (default 2)
$max_requests = 10;     # retire a child after that many accepts (default 10)
$child_timeout=5*60;    # abort child if it does not complete each task in n sec
                        # (default: 8*60 seconds)
```

61. **Setting up awstats with Postfix**
- Install the standard package '`awstats`'

- Make a copy of the main `awstats` configuration file:
```
cp  /etc/awstats/awstats.conf  /etc/awstats/awstats.mail.conf
```
- Comment out the include line at the end of this new file:
```
#Include "/etc/awstats/awstats.conf.local"
```

- Add the following lines in the end of this new file:

```
LogFile="perl /usr/share/doc/awstats/examples/maillogconvert.pl standard < /var/log/mail.log |"
LogType=M
LogFormat="%time2 %email %email_r %host %host_r %method %url %code %bytesd"
LevelForBrowsersDetection=0
LevelForOSDetection=0
LevelForRefererAnalyze=0
LevelForRobotsDetection=0
LevelForWormsDetection=0
LevelForSearchEnginesDetection=0
LevelForFileTypesDetection=0
ShowMenu=1
ShowSummary=HB
ShowMonthStats=HB
ShowDaysOfMonthStats=HB
ShowDaysOfWeekStats=HB
ShowHoursStats=HB
ShowDomainsStats=0
ShowHostsStats=HBL
ShowAuthenticatedUsers=0
ShowRobotsStats=0
ShowEMailSenders=HBML
ShowEMailReceivers=HBML
```

```
ShowSessionsStats=0
ShowPagesStats=0
ShowFileTypesStats=0
ShowFileSizesStats=0
ShowBrowsersStats=0
ShowOSStats=0
ShowOriginStats=0
ShowKeyphrasesStats=0
ShowKeywordsStats=0
ShowMiscStats=0
ShowHTTPErrorsStats=0
ShowSMTPErrorsStats=1
```

- Disable the cronjob in: (it was meant for the web server only)
  `/etc/cron.d/awstats`

- Create a cron job that will run regularly(eg. each hour) the following command:
  (The command is all on the same line)

```
perl /usr/lib/cgi-bin/awstats.pl -config=mail -update -output >
        /srv/www/mailstats/index.html
```

62. **Activate an init script with default runlevels:**

```
update-rc.d initscriptname defaults
```

63. **Correct the locales:**
Edit the file: `/etc/environment`
make sure the following elements are set:

```
LANG="de_DE.UTF-8"
LC_ALL="de_DE.UTF-8"
LANGUAGE="de_DE.UTF-8"
LC_TYPE="de_DE.UTF-8"
```

64.**Always get MC in english:**
enter the following alias in the `~/.bashrc` or `~/.profile`

```
alias mc='env LANGUAGE=en_US.UTF-8 mc'
```

65.**Verify integrity of all installed packages:**

```
debsums -ca
```

66.**VGA Resolution and Color Depth reference Chart**:

Depth	800×600	1024×768	1152×864	1280×1024	1600×1200
8 bit	vga=771	vga=773	vga=353	vga=775	vga=796
16 bit	vga=788	vga=791	vga=355	vga=794	vga=798
24 bit	vga=789	vga=792		vga=795	vga=799

67.**Create an SSL certificate for vsftpd**:

```
make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/ssl/certs/vsftpd.pem
```

68.**To repair the locales in system**

```
export  LANGUAGE=de_DE.UTF-8
export  LANG=de_DE
apt-get install -f
```

69. **Create an SSL self signed certificate for Apache2:**

Here is a step-by-step description:

1. Make sure OpenSSL is really installed and in your PATH. But some commands even work ok when you just run the ``openssl'' program from within the OpenSSL source tree as ``./apps/openssl''.

2. Create a RSA private key for your Apache server (will be Triple-DES encrypted and PEM formatted):

```
$ openssl genrsa -des3 -out server.key 1024
```

Please backup this server.key file and remember the pass-phrase you had to enter at a secure location. You can see the details of this RSA private key via the command:

```
$ openssl rsa -noout -text -in server.key
```

And you could create a decrypted PEM version (not recommended) of this RSA private key via:

```
$ openssl rsa -in server.key -out server.key.unsecure
```

3. Create a Certificate Signing Request (CSR) with the server RSA private key (output will be PEM formatted):

```
$ openssl req -new -key server.key -out server.csr
```

Make sure you enter the FQDN ("Fully Qualified Domain Name") of the server when OpenSSL prompts you for the "CommonName", i.e. when you generate a CSR for a website which will be later accessed via https://www.foo.dom/, enter "www.foo.dom" here. You can see the details of this CSR via the command

```
$ openssl req -noout -text -in server.csr
```

4. You now have to send this Certificate Signing Request (CSR) to a Certifying Authority (CA) for signing. The result is then a real Certificate which can be used for Apache. Here you have to options: First you can let the CSR sign by a commercial CA like Verisign or Thawte. Then you usually have to post the CSR into a web form, pay for the signing and await the signed Certificate you then can store into a server.crt file. For more information about commercial CAs have a look at the following locations:

* Verisign
  http://digitalid.verisign.com/server/apacheNotice.htm
* Thawte Consulting
  http://www.thawte.com/ssl-digital-certificates/buy-ssl-certificates
* CertiSign Certificadora Digital Ltda.
  http://www.certisign.com.br
* IKS GmbH
  http://www.iks-jena.de/leistungen/ca/
* Uptime Commerce Ltd.

* BelSign NV/SA, now GlobalSign
  http://www.globalsign.com/index.htm

Second you can use your own CA and now have to sign the CSR yourself by this CA. Read the next answer in this FAQ on how to sign a CSR with your CA yourself. You can see the details of the received Certificate via the command:

```
$ openssl x509 -noout -text -in server.crt
```

5. Now you have two files: server.key and server.crt. These now can be used as following inside your Apache's httpd.conf file:

```
SSLCertificateFile    /path/to/this/server.crt
SSLCertificateKeyFile /path/to/this/server.key
```

The `server.csr` file is no longer needed.

70.**How can I create and use my own Certificate Authority (CA)?**

The short answer is to use the CA.sh or CA.pl script provided by OpenSSL. The long and manual answer is this:

1. Create a RSA private key for your CA (will be Triple-DES encrypted and PEM formatted):

```
$ openssl genrsa -des3 -out ca.key 1024
```

Please backup this ca.key file and remember the pass-phrase you currently entered at a secure location. You can see the details of this RSA private key via the command

```
$ openssl rsa -noout -text -in ca.key
```

And you can create a decrypted PEM version (not recommended) of this private key via:

```
$ openssl rsa -in ca.key -out ca.key.unsecure
```

2. Create a self-signed CA Certificate (X509 structure) with the RSA key of the CA (output will be PEM formatted):

```
$ openssl req -new -x509 -days 365 -key ca.key -out ca.crt
```

You can see the details of this Certificate via the command:

```
$ openssl x509 -noout -text -in ca.crt
```

3. Prepare a script for signing which is needed because the ``openssl ca'' command has some strange requirements and the default OpenSSL config doesn't allow one easily to use ``openssl ca'' directly. So a script named sign.sh is distributed with the mod_ssl distribution (subdir pkg.contrib/). Use this script for signing.

4. Now you can use this CA to sign server CSR's in order to create real SSL Certificates for use inside an Apache webserver (assuming you already have a server.csr at hand):

```
$ ./sign.sh server.csr
```

This signs the server CSR and results in a server.crt file.

How can I change the pass-phrase on my private key file?

You simply have to read it with the old pass-phrase and write it again by specifying the new pass-phrase. You can accomplish this with the following commands:

```
$ openssl rsa -des3 -in server.key -out server.key.new
$ mv server.key.new server.key
```

Here you're asked two times for a PEM pass-phrase. At the first prompt enter the old pass-phrase and at the second prompt enter the new pass-phrase.

How can I get rid of the pass-phrase dialog at Apache startup time?

The reason why this dialog pops up at startup and every re-start is that the RSA private key inside your server.key file is stored in encrypted format for security reasons. The pass-phrase is needed to be able to read and parse this file. When you can be sure that your server is secure enough you perform two steps:

1. Remove the encryption from the RSA private key (while preserving the original file):

```
$ cp server.key server.key.org
$ openssl rsa -in server.key.org -out server.key
```

2. Make sure the server.key file is now only readable by root:

```
$ chmod 400 server.key
```

Now server.key will contain an unencrypted copy of the key. If you point your server at this file it will not prompt you for a pass-phrase. HOWEVER, if anyone gets this key they will be able to impersonate you on the net. PLEASE make sure that the permissions on that file are really such that only root or the web server user can read it (preferably get your web server to start as root but run as another server, and have the key readable only by root).

As an alternative approach you can use the ``SSLPassPhraseDialog exec:/path/to/program'' facility. But keep in mind that this is neither more nor less secure, of course.

How do I verify that a private key matches its Certificate?

The private key contains a series of numbers. Two of those numbers form the "public key", the others are part of your "private key". The "public key" bits are also embedded in your Certificate (we get them from your CSR). To check that the public key in your cert matches the public portion of your private key, you need to view the cert and the key and compare the numbers. To view the Certificate and the key run the commands:

```
$ openssl x509 -noout -text -in server.crt
$ openssl rsa -noout -text -in server.key
```

The `modulus' and the `public exponent' portions in the key and the Certificate must match. But since the public exponent is usually 65537 and it's bothering comparing long modulus you can use the following approach:

```
$ openssl x509 -noout -modulus -in server.crt | openssl md5
$ openssl rsa -noout -modulus -in server.key | openssl md5
```

And then compare these really shorter numbers. With overwhelming probability they will differ if the keys are different. BTW, if I want to check to which key or certificate a particular CSR belongs you can compute

```
$ openssl req -noout -modulus -in server.csr | openssl md5
```

71.Chapter 6 - Debian package management (long but useful)

`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`.

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 the method described in `chroot`, Section 8.6.35 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.

6.1 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` :-)

6.1.1 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. See Package dependencies, Section 2.2.8.

6.1.2 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
```

6.2 Beginning Debian package management

6.2.1 Set up APT

Set up `sources.list` as described in Preparing for upgrade, Section 5.2. [34] Also refer to Debian System installation hints, Chapter 3, Upgrading a distribution to `stable`, `testing`, or `unstable`, Chapter 5, and Rescue editors, Section 11.2.

6.2.2 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.

6.2.3 `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.

6.2.4 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.

6.2.5 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` 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`, Section 6.3.3.

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.

6.2.6 `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. [35]

- `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";
.
```

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

6.3 Debian survival commands

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

6.3.1 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
```

More support resources are listed at Support for Debian, Chapter 15.

6.3.2 APT upgrade troubleshooting

Package dependency problems may occur when upgrading in `unstable` or `testing` as described in Upgrading, Section 5.3. 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` and The `equivs` package, Section 6.5.2.

6.3.3 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.

6.3.4 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
```

6.3.5 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`, 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, Section 6.3.4. ([FIXME]: This procedure needs more experiments to verify.)

6.3.6 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. See Booting the system, Section 8.1. 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.)

6.3.7 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.

6.4 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**. :-)

6.4.1 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
```

6.4.2 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*
```

6.4.3 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, Section 6.4.4.

6.4.4 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. [36]

```
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
...
```

6.4.5 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 ...
```

6.4.6 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$"
```

6.4.7 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
```

6.4.8 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
```

6.4.9 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
```

6.4.10 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
```

6.4.11 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.

6.4.12 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.

6.4.13 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
```

6.4.14 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).

6.4.15 Optimized `sources.list`

In short, fancy efforts to create an optimized `sources.list` did not produce a significant improvement for me from a location in the USA. I manually chose a nearby site using `apt-setup`.

`apt-spy` creates `sources.list` automatically, based on latency and bandwidth. `netselect-apt` creates a more complete `sources.list`, but uses an inferior method of choosing the best mirror (ping time comparison).

```
# aptitude install apt-spy
# cd /etc/apt ; mv sources.list sources.list.org
# apt-spy -d testing -l sources.apt
```

6.5 Other Debian peculiarities

6.5.1 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 (see Preservation of local configuration, Section 2.2.4).

```
# 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.

6.5.2 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.
```

6.5.3 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`. For details, see Custom X sessions, Section 9.4.5.1.

`/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`.

6.5.4 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 as described in Customizing runlevels, Section 2.4.3. 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.

For more information about runlevels see Runlevels, Section 2.4.2.

6.5.5 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` as in Restricting logins with PAM, Section 9.2.1. The `pidof` command is also useful (see `pidof(8)`).