Debian Reference

Osamu Aoki `<debian@aokiconsulting.com>`
Editor: David Sewell `<dsewell@virginia.edu>`
'Authors' on page 173

CVS, Sun, 13 Oct 2002 22:40:09 -0600

Abstract

This Debian Reference (`http://qref.sourceforge.net/`) is intended to provide a broad overview of the Debian system as a **post-installation user's guide**. It covers many aspects of system administration through **shell-command** examples. Basic tutorials, tips, and other information are provided for topics including fundamental concepts of the Debian system, system installation hints, Debian package management, the Linux kernel under Debian, system tuning, building a gateway, text editors, CVS, programming, and GnuPG for **non-developers**.

Copyright Notice

Contents

Chapter 1

Preface

This Debian Reference (`http://qref.sourceforge.net/`) is intended to provide a broad overview of the Debian system as a **post-installation user's guide**. Its target reader is someone who is willing to read shell scripts. I expect the reader to have gained basic skills in Unix-like systems prior to reading this document.

I made a conscious decision **not** to explain everything in detail if it can be found in a **manual page**, an **info page**, or a **HOWTO document**. Instead of full explanations, I have tried to give more directly practical information by providing *exact command sequences* or *example scripts*. Much of the information included consists of reminders or pointers to the authoritative references listed in 'References' on page 165. This is partly because this document originated as a "**quick reference**".

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

For help with emergency system maintenance, proceed to 'Debian survival commands' on page 58 immediately.

1.1 Official document

The latest official document is in Debian archives with the package name `debian-reference` and is also available from `http://www.debian.org/doc/manuals/debian-reference/`.

The latest development version is `http://qref.sourceforge.net/Debian/`. The project is hosted at `http://qref.sourceforge.net/`, where this document is available for download in plain text, HTML, PDF, SGML and PostScript formats.

1.2 Document conventions

This “Debian Reference” document provides information through short Bash shell commands. Here are the conventions used:

```
# command in root account
$ command in user account
... description of action
```

See ‘Bash – **GNU** standard interactive shell’ on page 143 for more information on Bash.

Reference to:

- a **Unix manual** page is given in the form `bash(1)`.
- a **GNU TEXINFO** page is given in the form `info libc`.
- a **book** is given in the form *The C Programming Language*.
- a **URL** is given in the form `http://www.debian.org/doc/manuals/debian-reference/`.
- a **file** on the system is given in the form `/usr/share/doc/Debian/reference/`.

The following abbreviations are used:

- **LDP**: Linux Documentation Project (`http://www.tldp.org/`)
- **DDP**: Debian Documentation Project (`http://www.debian.org/doc/`)

In this document only URLs are shown for LDP documents, but they can also be obtained as a package and installed into `/usr/share/doc/HOWTO/`. See ‘References’ on page 165.

Sample scripts are available in the examples subdirectory (`examples/`); for hidden files, the preceding “.” in the filename is converted to underscore “_”.

1.3 Basic setup

If the system is installed with the bare minimum of packages, make sure to execute the following commands to install some essential packages and a few key documents:

```
# apt-get install info man-db doc-base dhelp apt apt-utils auto-apt \
                  dpkg less mc ssh nano-tiny elvis-tiny vim sash \
                  kernel-package \
                  manpages manpages-dev doc-debian doc-linux-text \
                  debian-policy developers-reference maint-guide \
                  apt-howto harden-doc install-doc \
                  libpam-doc glibc-doc samba-doc exim-doc cvsbook \
                  gnupg-doc
# apt-get install debian-reference # for Sarge, do this too :)
```

1.4 Basics of the Debian distributions

Debian comes in 3 release “flavors”:

- stable: Good to track on a production server. Boring for the workstation (WS). See ‘The `stable` distribution’ on page 6.
- testing: Good to track on the WS. See ‘The `testing` distribution’ on page 7.
- unstable: Never track this blindly. See ‘The `unstable` distribution’ on page 7.

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`. See ‘The Debian archives’ on page 5.

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

- ‘Install system with APT’ on page 58 (mainly for WS purposes)
- ‘Port a package to the `stable` system’ on page 67 (mainly for server purposes)

After explaining some fundamentals of the Debian distribution in ‘Debian fundamentals’ on page 5, I will present some basic information to help you live happily with the latest software, taking advantage of the `testing` and `unstable` distributions of Debian. The impatient should proceed to ‘Debian survival commands’ on page 58 immediately. Happy upgrading!

Chapter 2

Debian fundamentals

This chapter provides fundamental information on the Debian system for non-developers. For authoritative information, see:

- Debian Policy Manual
- Debian Packaging Manual (Potato)
- Debian Developer's Reference
- Debian New Maintainers' Guide

listed under 'References' on page 165.

If you are looking for less detailed "how-to" explanations, jump directly to 'Debian package management' on page 57 or other relevant chapters.

This chapter consists of documents taken from the "Debian FAQ", greatly reorganized to allow the ordinary Debian system administrator to get started.

2.1 The Debian archives

2.1.1 Directory structures

The software that has been packaged for Debian is available in one of several directory trees on each Debian mirror site (`http://www.debian.org/misc/README.mirrors`) through FTP or HTTP.

The following directories can be found on each Debian mirror site under the `/debian/` directory:

`/dists/`: This directory contains the "distributions", and this used to be the canonical way to access the currently available packages in Debian releases and pre-releases. Some old packages and `Packages.gz` files are still in here.

`/pool/`: The new physical location for all packages of Debian releases and pre-releases.

`/tools/`: DOS utilities for creating boot disks, partitioning your disk drive, compressing/decompressing files, and booting Linux.

`/doc/`: The basic Debian documentation, such as the FAQ, the bug reporting system instructions, etc.

`/indices/`: The Maintainers file and the override files.

`/project/`: mostly developer-only materials, such as:

> **`project/experimental/`:** This directory contains packages and tools which are still being developed, and are still in the alpha testing stage. Users shouldn't be using packages from here, because they can be dangerous and harmful even for the most experienced.
>
> **`project/orphaned/`:** Packages that have been orphaned by their old maintainers, and withdrawn from the distribution.

2.1.2 Debian distributions

Normally there are three Debian distributions in the `dists` directory. They are named the "stable" distribution, the "testing" distribution, and the "unstable" distribution. Sometimes there is also a "frozen" distribution. Each distribution is defined as a symlink to the actual directory with a codename in the `dists` directory.

2.1.3 The `stable` distribution

Package entries for the `stable` distribution, Debian Woody (3.0r0), are recorded into the `stable` (symlink to `Woody`) directory:

- `stable/main/`: This directory contains the packages which formally constitute the most recent release of the Debian system.

  These packages all comply with the Debian Free Software Guidelines (`http://www.debian.org/social_contract#guidelines`) (also available as `/usr/share/doc/debian/social-contra` installed by `debian-doc`), and are all freely usable and distributable.

- `stable/non-free/`: This directory contains packages distribution of which is restricted in a way that requires that distributors take careful account of the specified copyright requirements.

  For example, some packages have licenses which prohibit commercial distribution. Others can be redistributed but are in fact shareware and not free software. The licenses of each of these packages must be studied, and possibly negotiated, before the packages are included in any redistribution (e.g., in a CD-ROM).

- `stable/contrib/`: This directory contains packages which are DFSG-free and **freely distributable** themselves, but somehow depend on a package that is **not** freely distributable and thus available only in the non-free section.

Now, in addition to the above locations, new physical packages are located under the `pool` directory ('The `pool` directory' on page 9).

The current status of `stable` distribution bugs is reported on the Stable Problems (`http://ftp-master.debian.org/testing/stable_probs.html`) Web page.

2.1.4 The `testing` distribution

Package entries for the `testing` distribution, Debian Sarge, are recorded into the `testing` (symlink to `Sarge`) directory after they have undergone some degree of testing in `unstable`. Now, in addition to the above locations, new physical packages are located under the `pool` directory ('The `pool` directory' on page 9). There are also `main`, `contrib` and `non-free` subdirectories in `testing`, which serve the same functions as in `stable`.

These packages must be in sync on all architectures where they have been built and mustn't have dependencies that make them uninstallable; they also have to have fewer release-critical bugs than the versions currently in `unstable`. This way, we hope that `testing` is always close to being a release candidate. More details of the testing mechanism are at `http://ftp-master.debian.org/testing/`.

The latest status of the `testing` distribution is reported at these sites:

- update excuses (`http://ftp-master.debian.org/testing/update_excuses.html`)
- testing problems (`http://ftp-master.debian.org/testing/testing_probs.html`)
- release-critical bugs (`http://bugs.debian.org/release-critical/`)
- base system bugs (`http://base.debian.net/`)
- bugs in standard and task packages (`http://standard.debian.net/`)
- other bugs and bug-squashing party notes (`http://bugs.debian.net/`)

2.1.5 The `unstable` distribution

Package entries for the `unstable` distribution, `sid`, are recorded into the `unstable` (symlink to `sid`) directory after they are uploaded to the Debian archive and stay here until they are moved to `testing`. New physical packages are located under the `pool` directory ('The `pool` directory' on page 9). There are also `main`, `contrib` and `non-free` subdirectories in `unstable`, which serve the same functions as in `stable`.

The `unstable` distribution contains a snapshot of the most current development system. Users are welcome to use and test these packages, but are warned about their state of readiness. The advantage of using the `unstable` distribution is that you are always up-to-date with the latest in the Debian software project—but if it breaks, you get to keep both parts :-)

The current status of `unstable` distribution bugs is reported on the Unstable Problems (`http://ftp-master.debian.org/testing/unstable_probs.html`) Web page.

2.1.6 The **`frozen`** distribution

When the `testing` distribution is mature enough, it becomes frozen, meaning no new code is accepted anymore, just bugfixes, if necessary. Also, a new testing tree is created in the `dists` directory, assigned a new codename. The frozen distribution passes through a few months of testing, with intermittent updates and deep freezes called "test cycles". (The recent `woody` release process did not create a symbolic link `frozen`, so `frozen` was not a distribution but just a development stage of the `testing` distribution.)

We keep a record of bugs in the frozen distribution that can delay a package from being released or bugs that can hold back the whole release. Once that bug count lowers to maximum acceptable values, the frozen distribution becomes stable, it is released, and the previous stable distribution becomes obsolete (and moves to the archive).

2.1.7 Debian distribution codenames

Physical directory names in the `dists` directory, such as `Woody` and `Sarge`, are just "codenames". When a Debian distribution is in the development stage, it has no version number, but a codename instead. The purpose of these codenames is to make the mirroring of the Debian distributions easier (if a real directory like `unstable` suddenly changed its name to `stable`, a lot of stuff would have to be needlessly downloaded again).

Currently, `stable` is a symbolic link to `Woody`, and `testing` is a symbolic link to `Sarge`. This means that `Woody` is the current stable distribution and `Sarge` is the current testing distribution.

`unstable` is a permanent symbolic link to `sid`, as `sid` is always the unstable distribution.

2.1.8 Codenames used in the past

Other codenames that have already been used are: `buzz` for release 1.1, `rex` for release 1.2, `bo` for releases 1.3.x, `hamm` for release 2.0, `slink` for release 2.1, and `potato` for release 2.2.

2.1.9 The source for codenames

So far they have been characters taken from the movie *Toy Story* by Pixar.

- **Buzz** (Buzz Lightyear) was the spaceman,
- **Rex** was the tyrannosaurus,
- **Bo** (Bo Peep) was the girl who took care of the sheep,
- **Hamm** was the piggy bank,
- **Slink** (Slinky Dog) was the toy dog,
- **Sarge** was a leader of the Green Plastic Army Men,
- **Potato** was, of course, Mr. Potato Head,
- **Woody** was the cowboy.
- **Sid** was a boy next door who destroyed toys.

2.1.10 The `pool` directory

Historically, packages were kept in the subdirectory of `dists` corresponding to the distribution that contained them. This turned out to cause various problems, such as large bandwidth consumption on mirrors when major changes were made.

Packages are now kept in a large "pool", structured according to the name of the source package. To make this manageable, the pool is subdivided by section (`main`, `contrib` and `non-free`) and by the first letter of the source package name. These directories contain several files: the binary packages for each architecture, and the source packages from which the binary packages were generated.

You can find out where each package is placed by executing a command like `apt-cache showsrc` *`mypackagename`* and looking at the "Directory:" line. For example, the `apache` packages are stored in `pool/main/a/apache/`. Since there are so many `lib*` packages, these are treated specially: for instance, `libpaper` packages are stored in `pool/main/libp/libpaper/`.

The `dists` directories are still used for the index files used by programs like `apt`. Also, at the time of writing, older distributions have not been converted to use pools, so you'll see paths containing distribution names such as `potato` or `woody` in the "Filename" header field.

Normally, you won't have to worry about any of this, as new `apt` and probably older `dpkg-ftp` (see 'Methods for upgrading a Debian system' on page 19) will handle it seamlessly. If you want more information, see the Debian Package Pools FAQ (`http://people.debian.org/~joeyh/poolfaq`).

2.1.11 Historical notes about `sid`

When the present-day `sid` did not exist, the Debian archive site organization had one major flaw: there was an assumption that when an architecture was created in the current `unstable`, it would be released when that distribution became the new `stable`. For many architectures that wasn't the case, with the result that those directories had to be moved at release time. This was impractical because the move would chew up lots of bandwidth.

The archive administrators worked around this problem for several years by placing binaries for unreleased architectures in a special directory called `sid`. For those architectures not yet released, the first time they were released there was a link from the current `stable` to `sid`, and from then on they were created inside the `unstable` tree as usual. This layout was somewhat confusing to users.

With the advent of package pools (see 'The `pool` directory' on the page before) during the `woody` distribution development, binary packages began to be stored in a canonical location in the pool, regardless of the distribution, so releasing a distribution no longer causes large bandwidth consumption on the mirrors (there is, however, a lot of gradual bandwidth consumption throughout the development process).

2.1.12 Uploaded packages in `incoming`

Uploaded packages are first located at `http://incoming.debian.org/` after being checked to insure that they really come from a Debian developer (and are put in the `DELAYED` subdirectory in the case of a Non-Maintainer Upload (NMU)). Once a day, they are moved out of `incoming` to `unstable`.

In an emergency, you may want to install packages from `incoming` before they reach `unstable`.

2.1.13 Retrieving an older package

While the most recent Debian distributions are kept under the `debian` directory on each Debian mirror site (`http://www.debian.org/misc/README.mirrors`), archives for older Debian distributions such as Slink are kept on `http://archive.debian.org/` or under the `debian-archive` directory on each Debian mirror site.

Older `testing` and `unstable` packages can be located at `http://snapshot.debian.net/`.

2.1.14 Architecture sections

Within each of the major directory trees (`dists/stable/main`, `dists/stable/contrib`, `dists/stable/non-free`, `dists/unstable/main/`, etc.), the binary package entries reside in subdirectories whose names indicate the chip architecture for which they were compiled.

- `binary-all/`, for packages which are architecture-independent. These include, for example, Perl scripts, or pure documentation.

- `binary-`*`platform`*`/`, for packages which execute on a particular binary platform.

Please note that the actual binary packages for `testing` and `unstable` no longer reside in these directories, but in the top-level `pool` directory. The index files (`Packages` and `Packages.gz`) have been kept, though, for backwards compatibility.

For the actual binary architectures supported, see the Release Notes for each distribution. They can be located at the Release Notes sites for stable (`http://www.debian.org/releases/stable/releasenotes`) and testing (`http://www.debian.org/releases/testing/releasenotes`).

2.1.15 The source code

Source code is included for everything in the Debian system. Moreover, the license terms of most programs in the system **require** that source code be distributed along with the programs, or that an offer to provide the source code accompany the programs.

Normally the source code is distributed in the `source` directories, which are parallel to all the architecture-specific binary directories, or more recently in the `pool` directory (see 'The `pool` directory' on page 9). To retrieve the source code without having to be familiar with the structure of the Debian archive, try a command like `apt-get source` *`mypackagename`*.

Some packages, notably `pine`, are only available in a source package due to their licensing limitations. (Recently the `pine-tracker` package has been provided to facilitate Pine installation.) The procedures described in 'Port a package to the `stable` system' on page 67 and 'Packaging' on page 159 provide ways to build a package manually.

Source code may or may not be available for packages in the `contrib` and `non-free` directories, which are not formally part of the Debian system.

2.2 The Debian package management system

2.2.1 Overview of Debian packages

Packages generally contain all of the files necessary to implement a set of related commands or features. There are two types of Debian packages:

- **Binary packages**, which contain executables, configuration files, man/info pages, copyright information, and other documentation. These packages are distributed in a Debian-specific archive format (see 'Debian package format' on the next page); they are usually distinguished by having a `.deb` file extension. Binary packages can be unpacked using the Debian utility `dpkg`; details are given in its manual page.

- **Source packages**, which consist of a `.dsc` file describing the source package (including the names of the following files), a `.orig.tar.gz` file that contains the original unmodified source in gzip-compressed tar format, and usually a `.diff.gz` file that contains the Debian-specific changes to the original source. The utility `dpkg-source` packs and unpacks Debian source archives; details are provided in its manual page.

Installation of software by the package system uses "dependencies" which are carefully designed by the package maintainers. These dependencies are documented in the `control` file associated with each package. For example, the package containing the GNU C compiler (`gcc`) "depends" on the package `binutils` which includes the linker and assembler. If a user attempts to install `gcc` without having first installed `binutils`, the package management system (dpkg) will send an error message that it also needs `binutils`, and stop installing `gcc`. (However, this facility can be overridden by the insistent user; see `dpkg(8)`.) For additional details, see 'Package dependencies' on page 15 below.

Debian's packaging tools can be used to:

- manipulate and manage packages or parts of packages,
- aid the user in the splitting of packages that must be transmitted through a limited-size medium such as floppy disks,
- aid developers in the construction of package archives, and
- aid users in the installation of packages which reside on a remote Debian archive site.

2.2.2 Debian package format

A Debian "package", or a Debian archive file, contains the executable files, libraries, and documentation associated with a particular program suite or set of related programs. Normally, a Debian archive file has a filename that ends in `.deb`.

The internals of this Debian binary package format are described in the `deb(5)` manual page. Because this internal format is subject to change (between major releases of Debian), always use `dpkg-deb(8)` for manipulating `.deb` files.

Through at least the Woody distribution, all Debian archive files have been manipulable by the standard Unix commands `ar` and `tar`, even when dpkg commands are not available.

2.2.3 Naming conventions for Debian package filenames

The Debian package filenames conform to the following convention:

```
foo_VersionNumber-DebianRevisionNumber.deb
```

where *foo* represents the package name. As a check, one can determine the package name associated with a particular Debian archive file (`.deb` file) in one of these ways:

- inspect the "Packages" file in the directory where it was stored at a Debian archive site. This file contains a stanza describing each package; the first field in each stanza is the formal package name.
- use the command `dpkg --info foo_VVV-RRR.deb` (where *VVV* and *RRR* are the version and revision of the package in question, respectively). This displays, among other things, the package name corresponding to the archive file being unpacked.

The *VVV* component is the version number specified by the upstream developer. There are no standards governing version numbers, so they may have formats as different as "19990513" and "1.3.8pre1".

The *RRR* component is the Debian revision number, and is specified by the Debian developer (or an individual user if he chooses to build the package himself). This number corresponds to the revision level of the Debian package; thus, a new revision level usually signifies changes in the Debian makefile (`debian/rules`), the Debian control file (`debian/control`), the installation or removal scripts (`debian/p*`), or in the configuration files used with the package.

2.2.4 Preservation of the local configuration

Preservation of user-configurable files is enabled through Debian's "conffiles" mechanism. User configuration files (usually placed in `/etc`) are specified in the `conffiles` within the Debian package system. The package management system guarantees not to overwrite these files when the package is upgraded.

When it is possible to configure the system without modifying files that belong to various Debian packages, it is usually a good idea not to modify them even if they are "conffiles". This ensures faster and smoother upgrade operations.

To determine exactly which files are preserved during an upgrade, run:

```
dpkg --status package
```

and look under "Conffiles:".

Specifics regarding the contents of a Debian `conffiles` file are provided in the Debian Policy Manual, section 11.7 (see 'References' on page 165).

2.2.5 Debian maintenance scripts

Debian maintenance scripts are executable scripts which are automatically run before or after a package is installed. Along with a file named `control`, all of these files are part of the "control" section of a Debian archive file.

The individual files are:

preinst This script executes before its package is unpacked from its Debian archive (`.deb`) file. Many "preinst" scripts stop services for packages which are being upgraded until their installation or upgrade is completed (following the successful execution of the "postinst" script).

postinst This script typically completes any required configuration of a package once it has been unpacked from its Debian archive (`.deb`) file. Often, "postinst" scripts ask the user for input, and/or warn the user that if he accepts default values, he should remember to go back and reconfigure the package as the situation warrants. Many "postinst" scripts then execute any commands necessary to start or restart a service once a new package has been installed or upgraded.

prerm This script typically stops any daemons which are associated with a package. It is executed before the removal of files associated with the package.

postrm This script typically modifies links or other files associated with a package, and/or removes files created by it. (Also see 'Virtual packages' on the facing page.)

Currently all of the control files can be found in the directory `/var/lib/dpkg/info`. The files relevant to package `foo` begin with the name "foo" and have file extensions of "preinst", "postinst", etc., as appropriate. The file `foo.list` in that directory lists all of the files that were installed with the package `foo`. (Note that the location of these files is a dpkg internal, and may be subject to change.)

2.2.6 Package priorities

Each Debian package is assigned a **priority** by the distribution maintainers, as an aid to the package management system. The priorities are:

- **Required** packages are necessary for the proper functioning of the system.

  This includes all tools that are necessary to repair system defects. You must not remove these packages or your system may become totally broken and you may not even be able to use `dpkg` to put restore things. Systems with only the Required packages are probably inadequate for most purposes, but they do have enough functionality to allow the sysadmin to boot and install more software.

- **Important** packages should be found on any Unix-like system.

  Other packages without which the system will not run well or be usable will carry this priority. This does **not** include Emacs or X11 or TeX or any other large applications. These packages only constitute the bare infrastructure.

- **Standard** packages are standard on any Linux system, including a reasonably small but not too limited character-mode system.

  This is what will install by default if users do not select anything else. “Standard” does not include many large applications, but it does include Emacs (this is more a piece of infrastructure than an application) and a reasonable subset of TeX and LaTeX (if this turns out to be possible without X).

- **Optional** packages include all those that you might reasonably want to install even if you are unfamiliar with them, and if you don’t have have specialized requirements.

  This includes X11, a full TeX distribution, and lots of applications.

- **Extra** packages either conflict with others with higher priorities, are only likely to be useful if you already know what they are, or have specialized requirements that make them unsuitable for “Optional”.

2.2.7 Virtual packages

A virtual package is a generic name that applies to any one of a group of packages, all of which provide similar basic functionality. For example, both the `tin` and `trn` programs are news readers, and either one should therefore satisfy any dependency of a program that requires a news reader on a system in order to work or to be useful. They are therefore both said to provide the “virtual package” called `news-reader`.

Similarly, `exim` and `sendmail` both provide the functionality of a mail transport agent. They are therefore said to provide the virtual package “mail transport agent”. If either one is installed, then any program depending on the installation of a `mail-transport-agent` will be satisfied by the existence of this virtual package.

Debian has a mechanism so that, if more than one package which provides the same virtual package is installed on a system, the system administrator can set one as the preferred package. The relevant command is `update-alternatives`, and is described further in ‘Alternative commands’ on page 70.

2.2.8 Package dependencies

The Debian package system has a range of package “dependencies” which are designed to indicate (in a single flag) the level at which Program A can operate independently of the existence of Program B on a given system:

- Package A **depends** on Package B if B absolutely must be installed in order to run A. In some cases, A depends not only on B, but on a specific version of B. In this case, the version dependency is usually a lower limit, in the sense that A depends on any version of B more recent than some specified version.
- Package A **recommends** Package B if the package maintainer judges that most users would not want A without also having the functionality provided by B.
- Package A **suggests** Package B if B contains files that are related to (and usually enhance) the functionality of A.
- Package A **conflicts** with Package B when A will not operate if B is installed on the system. Most often, conflicts are cases where A contains files which are an improvement over those in B. "Conflicts" status is often combined with "replaces".
- Package A **replaces** Package B when files installed by B are removed and (in some cases) overwritten by files in A.
- Package A **provides** Package B when all of the files and functionality of B are incorporated into A. This mechanism provides a way for users with constrained disk space to get only that part of package A which they really need.

More detailed information on the use of each these terms can be found in the Packaging Manual and the Policy Manual.

Note that `dselect` has more fine-grained control over packages specified by **recommends** and **suggests** than `apt-get`, which simply pulls all the packages specified by **depends** and leaves all the packages specified by **recommends** and **suggests**. Both programs in modern form use APT as their back end.

2.2.9 The meaning of "pre-depends"

"Pre-depends" is a special dependency. In the case of an ordinary package, `dpkg` will unpack its archive file (i.e., its `.deb` file) independently of whether or not the files on which it depends exist on the system. Unpacking basically means that `dpkg` will extract the files from the archive file that were meant to be installed on your file system, and put them in place. If those packages **depend** on the existence of some other packages on your system, `dpkg` will refuse to complete the installation (by executing its "configure" action) until the other packages are installed.

However, there are some packages that `dpkg` will refuse even to unpack until certain dependencies are resolved. Such packages are said to "pre-depend" on the presence of some other package(s). The Debian project provided this mechanism to support the safe upgrading of systems from `a.out` format to ELF format, where the **order** in which packages were unpacked was critical. There are other large upgrade situations where this method is useful, e.g., for packages with “required” priority and their libc dependency.

Once again, more detailed information about this can be found in the Packaging Manual.

2.2.10 Package status

Package status can be “unknown”, “install”, “remove”, “purge”, or “hold”. These “want” flags indicate what the user wanted to do with a package (either by making choices in the “Select” section of `dselect`, or by directly invoking `dpkg`).

Their meanings are:

- **unknown** - the user has never indicated whether he wants the package.
- **install** - the user wants the package installed or upgraded.
- **remove** - the user wants the package removed, but does not want to remove any existing configuration files.
- **purge** - the user wants the package to be removed completely, including its configuration files.
- **hold** - the user wants this package not to be processed, i.e., he wants to keep the current version with the current status, whatever that is.

2.2.11 Holding back packages from an upgrade

There are two mechanisms for holding back packages from an upgrade, through `dpkg`, or, in Woody, through APT.

With `dpkg`, first export the list of package selections:

```
dpkg --get-selections \* > selections.txt
```

Then edit the resulting file *`selections.txt`*, changing the line containing the package you wish to hold, e.g. `libc6`, from this:

```
libc6                          install
```

to this:

```
libc6                          hold
```

Save the file, and reload it into `dpkg` database with:

```
dpkg --set-selections < selections.txt
```

Or, if you know the package name to hold, simply run:

```
echo libc6 hold | dpkg --set-selections
```

This procedure holds packages at the install process of each package file.

The same effect can be obtained through `dselect`. Simply enter the [S]elect screen, find the package you wish to hold in its present state, and press the '=' key (or 'H'). The changes will take effect immediately after you exit the [S]elect screen.

The APT system in the Woody distribution has a new alternative mechanism for holding packages during the archive retrieval process using `Pin-Priority`. See the manual page `apt_preferences(5)`, along with `http://www.debian.org/doc/manuals/apt-howto/` or the `apt-howto` package.

2.2.12 Source packages

Source packages are distributed in a directory called `source`, and you can either download them manually, or use

```
apt-get source foo
```

to fetch them (see the `apt-get(8)` manual page on how to set up APT for doing that).

2.2.13 Building binary packages from a source package

For a package *foo*, you will need all of `foo_*.dsc`, `foo_*.tar.gz` and `foo_*.diff.gz` to compile the source (note: there is no `.diff.gz` for a Debian native package).

Once you have them, if you have the `dpkg-dev` package installed, the command

```
$ dpkg-source -x foo_version-revision.dsc
```

will extract the package into a directory called `foo-version`.

If you want just to compile the package, you may cd into the `foo-version` directory and issue the command

```
$ fakeroot debian/rules build
```

to build the program (you may need to install the `fakeroot` package first), then

```
$ fakeroot debian/rules binary
```

to build the package as root, and then

```
# su -c "dpkg -i ../foo_version-revision_arch.deb"
```

to install the newly built package. See 'Port a package to the `stable` system' on page 67.

2.2.14 Creating new Debian packages

For detailed information on creating new packages, read the New Maintainers' Guide, available in the `maint-guide` package, or at `http://www.debian.org/doc/manuals/maint-guide/`.

2.3 Upgrading a Debian system

One of Debian's goals is to provide a consistent upgrade path and a secure upgrade process, and we always do our best to make a new release smoothly upgradable from the previous ones. Packages will alert the user when there are important notices during the upgrade process, and will often provide a solution to a possible problem.

You should also read the Release Notes, the document that describes the details of specific upgrades, shipped on all Debian CDs, and available on the WWW at `http://www.debian.org/releases/stable/releasenotes` or `http://www.debian.org/releases/testing/releasenotes`.

A practical guide to upgrades is provided in 'Debian package management' on page 57. This section describes the fundamental details.

2.3.1 Methods for upgrading a Debian system

One can always simply execute an anonymous FTP or `wget` call to a Debian archive, peruse the directories until one finds a desired file, fetch it, and finally install it using `dpkg`. (Note that `dpkg` will install upgrade files in place, even on a running system.) Sometimes, however, a revised package will require the installation of a newly revised version of another package, in which case the installation will fail until/unless the other package is installed.

Many people find this manual approach much too time-consuming, since Debian evolves so quickly — typically, a dozen or more new packages are uploaded every week. This number is larger just before a new major release. To deal with this avalanche, many people prefer to use an automated program for upgrading. Several specialized package management tools are available for this purpose.

2.3.2 Package management tools overview

The Debian package management system has two objectives: the manipulation of the package file itself and the retrieval of package files from the Debian archive. `dpkg` performs the former task, APT and `dselect` the latter.

2.3.3 `dpkg`

This is the main program for manipulating package files; read `dpkg(8)` for a full description.

`dpkg` comes with several primitive supplemental programs.

- dpkg-deb: Manipulate `.deb` files. `dpkg-deb(1)`
- dpkg-ftp: An older package file retrieval command. `dpkg-ftp(1)`
- dpkg-mountable: An older package file retrieval command. `dpkg-mountable(1)`
- dpkg-split: Splits a large package into smaller files. `dpkg-split(1)`

`dpkg-ftp` and `dpkg-mountable` have been superseded by the introduction of the APT system.

2.3.4 APT

APT (the Advanced Packaging Tool) is an advanced interface to the Debian packaging system consisting of several programs whose names typically begin with “apt-”. `apt-get`, `apt-cache` and `apt-cdrom` are the command-line tools for handling packages. These also function as the user’s “back-end” programs to other tools, such as `dselect` and `aptitude`.

For more information, install the `apt` package and read `apt-get(8)`, `apt-cache(8)`, `apt-cdrom(8)`, `apt.conf(5)`, `sources.list(5)`, `apt_preferences(5)` (woody), and `/usr/share/doc/apt/guide.html/index.html`.

An alternative source of information is the APT HOWTO (`http://www.debian.org/doc/manuals/apt-howto/`). This can be installed by `apt-howto` at `/usr/share/doc/apt-howto/en/apt-howto-en.html/index.html`.

`apt-get upgrade` and `apt-get dist-upgrade` pull only the packages listed under "Depends:" and overlook all the packages listed under "Recommends:" and "Suggests:". To avoid this, use `dselect`.

2.3.5 `dselect`

This program is a menu-driven user interface to the Debian package management system. It is particularly useful for first-time installations and large-scale upgrades. See '`dselect` – global configuration' on page 66.

For more information, install the `install-doc` package and read `/usr/share/doc/install-doc/dselect-beginner.en.html` or dselect Documentation for Beginners (`http://www.debian.org/releases/woody/i386/dselect-beginner`).

2.3.6 Upgrading a running system

The kernel (file system) in Debian systems supports replacing files even while they're being used.

We also provide a program called `start-stop-daemon` which is used to start daemons at boot time or to stop daemons when the kernel runlevel is changed (e.g., from multi-user to single-user or to "halt"). The same program is used by installation scripts when a new package containing a daemon is installed, to stop running daemons, and to restart them as necessary.

Note that the Debian system does not require use of the single-user mode to upgrade a running system.

2.3.7 Downloaded and cached `.deb` archive files

If you have manually downloaded package files to your disk (which is not absolutely necessary, see above for the description of `dpkg-ftp` or APT), then after you have installed the packages, you can remove the `.deb` files from your system.

If APT is used, these files are cached in the `/var/cache/apt/archives/` directory. You may erase them after installation (`apt-get clean`) or copy them to another machine's `/var/cache/apt/archives/` directory to save downloading during subsequent installations.

2.3.8 Record-keeping for upgrades

`dpkg` keeps a record of the packages that have been unpacked, configured, removed, and/or purged, but does not (currently) keep a log of terminal activity that occurred while a package was being so manipulated.

The simplest way to work around this is to run your `dpkg`, `dselect`, `apt-get`, etc., sessions within the `script(1)` program.

2.4 The Debian boot process

2.4.1 The `init` program

Like all Unices, Debian boots up by executing the program `init`. The configuration file for `init` (which is `/etc/inittab`) specifies that the first script to be executed should be `/etc/init.d/rcS`. This script runs all of the scripts in `/etc/rcS.d/` by sourcing or forking a subprocess depending on their file extension to perform initialization such as checking and mounting file systems, loading modules, starting the network services, setting the clock, and performing other initialization. Then, for compatibility, it also runs the files (except those with a '.' in the filename) in `/etc/rc.boot/`. Any scripts in the latter directory are usually reserved for system administrator use, and using them in packages is deprecated. See 'System initialization hints' on page 97 for more info.

2.4.2 Runlevels

After completing the boot process, `init` executes all start scripts in a directory specified by the default runlevel (this runlevel is given by the entry for `id` in `/etc/inittab`). Like most System V compatible Unices, Linux has 7 runlevels:

- 0 (halt the system),
- 1 (single-user mode),
- 2 through 5 (various multi-user modes), and
- 6 (reboot the system).

Debian systems come with `id=2`, which indicates that the default runlevel will be 2 when the multi-user state is entered, and the scripts in `/etc/rc2.d/` will be run.

In fact, the scripts in any of the directories /etc/rc*N*.d/ are just symbolic links back to scripts in `/etc/init.d/`. However, the **names** of the files in each of the /etc/rc*N*.d/ directories are selected to indicate the **way** the scripts in `/etc/init.d/` will be run. Specifically, before entering any runlevel, all the scripts beginning with 'K' are run; these scripts kill services. Then all the scripts beginning with 'S' are run; these scripts start services. The two-digit number following the 'K' or 'S' indicates the order in which the script is run. Lower-numbered scripts are executed first.

This approach works because the scripts in `/etc/init.d/` all take an argument which can be either "start", "stop", "reload", "restart" or "force-reload" and will then do the task indicated by the argument. These scripts can be used even after a system has been booted, to control various processes.

For example, with the argument “reload” the command

```
# /etc/init.d/sendmail reload
```

sends the sendmail daemon a signal to reread its configuration file.

2.4.3 Customizing the boot process

Debian does not use a BSD-style `rc.local` directory to customize the boot process; instead it provides the following mechanism for customization.

Suppose a system needs to execute script `foo` on start-up, or on entry to a particular (System V) runlevel. Then the system administrator should:

1. Enter the script `foo` into the directory `/etc/init.d/`.
2. Run the Debian command `update-rc.d` with appropriate arguments, to set up links between the (command-line-specified) directories `rc?.d` and `/etc/init.d/foo`. Here, *?* is a number from 0 through 6 that corresponds to one of the System V runlevels.
3. Reboot the system.

The command `update-rc.d` will set up links between files in the directories `rc?.d` and the script in `/etc/init.d/`. Each link will begin with an ‘S’ or a ‘K’, followed by a number, followed by the name of the script. Scripts beginning with ‘S’ in `/etc/rcN.d/` are executed when runlevel *N* is entered. Scripts beginning with a ‘K’ are executed when leaving runlevel *N*.

One might, for example, cause the script `foo` to execute at boot-up, by putting it in `/etc/init.d/` and installing the links with `update-rc.d foo defaults 19`. The argument `defaults` refers to the default runlevels, which are 2 through 5. The argument `19` ensures that `foo` is called before any scripts containing numbers 20 or larger.

2.5 Supporting diversity

Debian offers several avenues to accommodate any wishes of the system administrator without breaking the system.

- `dpkg-divert`, see ‘The `dpkg-divert` command’ on page 70.
- `equivs`, see ‘The `equivs` package’ on page 70.

- `update-alternative`, see 'Alternative commands' on page 70.
- `make-kpkg` can accommodate many boot loaders. See `make-kpkg(1)` and 'Debian standard method' on page 73.

Any files under `/usr/local/` belong to the system administrator and Debian will not touch them. Most (or all) files under `/etc` are `conffiles` and Debian will not overwrite them upon upgrade unless the system administrator requests so explicitly.

2.6 Internationalization

The Debian system is internationalized and provides support for character display and entry in many languages, both within the console and under X. Many documents, manual pages, and system messages have been translated into a growing number of languages. During installation, Debian prompts the user to choose an installation language (and sometimes a local language variant).

If your installed system does not support all the language features you need, or if you need to change languages or install a different keyboard to support your language, see 'Localization and national language support' on page 119.

2.7 Debian and the kernel

See 'The Linux kernel under Debian' on page 73.

2.7.1 Compiling a kernel from non-Debian source

One has to understand the Debian policy with respect to headers.

The Debian C libraries are built with the most recent **stable** releases of the **kernel** headers.

For example, the Debian-1.2 release used version 5.4.13 of the headers. This practice contrasts with the Linux kernel source packages distributed at all Linux FTP archive sites, which use even more recent versions of the headers. The kernel headers distributed with the kernel source are located in `/usr/include/linux/include/`.

If you need to compile a program with kernel headers that are newer than those provided by `libc6-dev`, then you must add `-I/usr/src/linux/include/` to your command line when compiling. This came up at one point, for example, with the packaging of the automounter daemon (`amd`). When new kernels changed some internals dealing with NFS, `amd` needed to know about them. This required the inclusion of the latest kernel headers.

2.7.2 Tools to build custom kernels

Users who wish to (or must) build a custom kernel are encouraged to download the package `kernel-package`. This package contains the script to build the kernel package, and provides the capability to create a Debian kernel-image package just by running the command

```
# make-kpkg kernel_image
```

in the top-level kernel source directory. Help is available by executing the command

```
# make-kpkg --help
```

and through the manual page `make-kpkg(8)` and 'The Linux kernel under Debian' on page 73.

Users must separately download the source code for the most recent kernel (or the kernel of their choice) from their favorite Linux archive site, unless a kernel-source-*version* package is available (where *version* stands for the kernel version). The Debian `initrd` boot script requires a special kernel patch called `initrd`; see `http://bugs.debian.org/149236`.

Detailed instructions for using the `kernel-package` package are given in the file `/usr/doc/kernel-package`

2.7.3 Alternative boot loaders

To employ alternative boot loaders such as `grub` or `loadlin`, copy the compiled Linux kernel `bzimage` to other locations (e.g., to `/boot/grub` or to an MS-DOS partition).

2.7.4 Custom boot floppies

The task of making a custom boot floppy is greatly aided by the Debian package `boot-floppies`, normally found in the `admin` section of the Debian FTP archive. Shell scripts in this package produce boot floppies in `syslinux` format. These are MS-DOS formatted floppies whose master boot records have been altered so that they directly boot Linux (or whatever other operating system has been defined in the `syslinux.cfg` file on the floppy). Other scripts in this package produce emergency root disks and can even reproduce the base disks.

You will find more information about this in the `/usr/doc/boot-floppies/README` file after installing the `boot-floppies` package.

2.7.5 Special provisions for dealing with modules

Debian's `modconf` package provides a shell script (`/usr/sbin/modconf`) which can be used to customize the configuration of modules. This script presents a menu-based interface, prompting the user for particulars on the loadable device drivers in his system. The responses are used to customize the file `/etc/modules.conf` (which lists aliases, and other arguments that must be used in conjunction with various modules) through files in `/etc/modutils/`, and `/etc/modules` (which lists the modules that must be loaded at boot time).

Like the (new) Configure.help files that are now available to support the construction of custom kernels, the `modconf` package comes with a series of help files (in `/usr/lib/modules_help/`) which provide detailed information on appropriate arguments for each of the modules. See 'The modularized 2.4 kernel' on page 75 for examples.

2.7.6 De-installing an old kernel package

The `kernel-image-NNN.prerm` script checks to see whether the kernel you are currently running is the same as the kernel you are trying to de-install. Therefore you can safely remove unwanted kernel image packages using this command:

```
dpkg --purge --force-remove-essential kernel-image-NNN
```

(Replace *NNN* with your kernel version and revision number, of course.)

Chapter 3

Debian System installation hints

Official documentation for installing Debian is located at `http://www.debian.org/releases/stable/`, and `http://www.debian.org/releases/stable/installmanual`.

The development versions are located at `http://www.debian.org/releases/testing/`, and `http://www.debian.org/releases/testing/installmanual` (work in progress, sometimes this may not exist).

Although “Debian Reference” was written during the days of the Potato release, most of its contents have been updated to Debian Woody (3.0r0) and Debian Sarge.

3.1 General Linux system installation hints

In order to minimize risks associated with “testing” and “unstable” packages, it is a good practice to set up your main Linux system for dual booting along with another small stable Linux system.

3.1.1 Hardware compatibility basics

Linux is compatible with most PC hardware and can be installed to almost any system. For me it was as easy as installing Windows 95/98/Me. The hardware compatibility list just seems to keep growing.

If you have a laptop PC, check Linux on Laptops (`http://www.linux-laptop.net/`) for installation pointers by brand and model.

My recommendation for desktop PC hardware is “Just be conservative”:

- SCSI rather than IDE for work, IDE/ATAPI HD for private use.
- IDE/ATAPI CD-ROM (or CD-RW).

- PCI rather than ISA, especially for the network card (NIC).
- Use a cheap NIC. Tulip for PCI, NE2000 for ISA are good.
- Avoid PCMCIA (notebook) as your first Linux install.
- No USB keyboard, mouse . . . unless you want a challenge.

If you have a slow machine, yanking out the hard drive and plugging it into another faster machine for installation is a good idea.

3.1.2 Determining a PC's hardware and chip set

During installation, one will be asked to identify the hardware or chip set of the PC. Sometimes that information may not seem easy to find. Here is one method:

1. Open your PC's case and look inside.
2. Record the numbers on the large chips on the graphics card, network card, chip near serial ports, chip near IDE ports.
3. Record card names printed on the back of the PCI and ISA cards.

3.1.3 Determining a PC's hardware via Debian

The following commands on a Linux system should give some idea of actual hardware and configuration.

```
$ /sbin/lspci -v |pager
$ pager /proc/pci
$ pager /proc/interrupts
$ pager /proc/ioports
```

These commands can be run during the install process from the console screen by pressing ALT-F2.

3.1.4 Determining a PC's hardware via other OSs

Hardware information can also be obtained from other OSs.

Install another commercial Linux distribution. Hardware detection on those tends to be better than on Debian as of now. This may change as Woody evolves.

Install Windows. Hardware configuration can be obtained by right-clicking "My Computer" to get to Properties / Device Manager. Record all resource information such as IRQ, I/O port address, and DMA. Some old ISA cards may need to be configured under DOS and used accordingly.

3.1.5 A Lilo myth

Lilo is limited to 1024 cylinders. —WRONG !

The newer `lilo` used after Debian Potato has lba32 support. If the BIOS of your motherboard is recent enough to support lba32, `lilo` should be able to load beyond the old 1024-cylinder limitation.

Just make sure to add a line reading “lba32” somewhere near the beginning of your `lilo.conf` file if you have kept an old `lilo.conf`.

3.1.6 Choice of boot floppies

For Potato, I liked the IDEPCI disk set for normal install to a desktop. For Woody, I like the bf2.4 boot disk set. They both use a version of `boot-floppies` to create boot floppies.

If you have a PCMCIA network card, you need to use the standard boot disk set (largest number of floppies but all driver modules available) and configure the NIC in the PCMCIA setup; do not try to set up an NIC card in the standard network setup dialogue.

For special systems, you may need to create a custom rescue disk. This can be done by replacing the kernel image named “linux” on the Debian rescue disk by overwriting it with another compressed kernel image compiled off-site for the machine. Details are documented in `readme.txt` on the rescue disk. The rescue floppy uses the MS-DOS file system, so you can use any system to read and edit it. This should make life easier for people with a special network card, etc.

For Sarge, `debian-installer` and/or `pgi` is expected to be used for creating boot floppies.

3.1.7 Installation

Follow the official instructions found in `http://www.debian.org/releases/stable/installmanual` or `http://www.debian.org/releases/testing/installmanual` (work in progress, sometimes this may not exist).

If you are installing a system using boot floppies in the testing distribution, you may need to open a console terminal during the install process by pressing `ALT-F2` and manually edit `/etc/sources.list` entries from `stable` to `testing` to adjust APT sources.

I tend to install `lilo` into places like `/dev/hda3`, while installing `mbr` into `/dev/hda`. This minimizes the risk of overwriting boot information.

Here is what I choose during the install process.

- MD5 passwords “yes”
- shadow passwords “yes”
- Install “advanced” (dselect **) and select

  - Exclude emacs (if selected), nvi, tex, telnet, talk(d);
  - Include mc, vim, either one of nano-tiny or elvis-tiny. See '`dselect` – global configuration' on page 66. Even if you are an Emacs fan, avoid it now and be content with nano during install. Also avoid installing other large packages such as TEX (Potato used to do this) at this stage. See 'Rescue editors' on page 129 for the reason for installing nano-tiny or elvis-tiny here.
- All configuration questions = "y" (replace current) during each package install dialog.
- `exim`: select 2 for machine since I send mail through my ISP's SMTP server.

For more information on dselect, see '`dselect` – global configuration' on page 66.

3.1.8 Hosts and IP to use for LAN

Example of LAN configuration (C subnet: 192.168.1.0/24):

```
Internet
   |
   +--- External ISP provides POP service (accessed by fetchmail)
   |
  Access point ISP provides DHCP service and SMTP relay service
   |                      :
  Cable modem          (Dial-up)
   |                      :
LAN Gateway machine external port: eth0 (IP given by ISP's DHCP)
 use old notebook PC (IBM Thinkpad, 486 DX2 50 MHz, 20 MB RAM)
 run Linux 2.4 kernel with ext3 file system.
 run "ipmasq" package (with stronger patch, NAT and firewall)
 run "dhcp-client" package configured for eth0 (override DNS setting)
 run "dhcp" package configured for eth1
 run "exim" as the smarthost (mode 2)
 run "fetchmail" with a long interval (fallback)
 run "bind" as the cache nameserver for Internet from LAN
            as authoritative nameserver for LAN domain from LAN
 run "ssh" on port 22 and 8080 (connect from anywhere)
 run "squid" as the cache server for the Debian archive (for APT)
LAN Gateway machine internal port: eth1 (IP = 192.168.1.1, fixed)
                          |
         +--- LAN Switch (10 base T) ---+
         |                              |
Some fixed IP clients on LAN      Some DHCP clients on LAN
(IP = 192.168.1.2-127, fixed)     (IP = 192.168.1.128-200, dynamic)
```

See 'Building a gateway with a Debian system' on page 123 for the details of configuring the LAN gateway server.

3.1.9 User accounts

In order to have a consistent feel across machines, the first few accounts are always the same in my system.

I always create a first user account with a name like "admin" (uid=1000). I forward all root email there. This account is given membership in the `adm` group (see "'Why GNU `su` does not support the `wheel` group"' on page 99), which can be given a good amount of root privilege through `su` using PAM or the `sudo` command. See 'Add a user account' on page 46 for details.

3.1.10 Creating file systems

Hard disk partition

I prefer to use different partitions for different directory trees to limit damage upon system crash. E.g.,

```
/           == (/ + /boot + /bin + /sbin)
            == 50MB+
/tmp        == 100MB+
/var        == 100MB+
/home       == 100MB+
/usr        == 700MB+ with X
/usr/local  == 100MB
```

The size of the `/usr` directory is very dependent on X-window applications and documentation. `/usr` can be 300MB if one runs a console terminal only, whereas 2GB–3GB is not an unusual size if one has installed many Gnome applications. When `/usr` grows too big, moving out `/usr/share/` to a different partition is the most effective cure. With the new large prepackaged Linux 2.4 kernels, / may need more than 200MB.

For example, the current status of my Internet gateway machine is as follows (output of the `df -h` command):

```
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda3             300M  106M  179M  38% /
/dev/hda7             100M   12M   82M  13% /home
```

```
/dev/hda8            596M   53M  513M  10% /var
/dev/hda6            100M  834k   94M   1% /var/lib/cvs
/dev/hda9            596M  222M  343M  40% /usr
/dev/hda10           596M  130M  436M  23% /var/cache/apt/archives
/dev/hda11           1.5G  204M  1.2G  14% /var/spool/squid
```

(The large area reserved for `/var/spool/squid` is for a proxy cache for package downloading.)

Following is `fdisk -l` output to provide an idea of partition structure:

```
# fdisk -l /dev/hda # comment

/dev/hda1             1        41    309928+   6  FAT16 # DOS
/dev/hda2            42        84    325080   83  Linux # (not used)
/dev/hda3   *        85       126    317520   83  Linux # Main
/dev/hda4           127       629   3802680    5  Extended
/dev/hda5           127       143    128488+  82  Linux swap
/dev/hda6           144       157    105808+  83  Linux
/dev/hda7           158       171    105808+  83  Linux
/dev/hda8           172       253    619888+  83  Linux
/dev/hda9           254       335    619888+  83  Linux
/dev/hda10          336       417    619888+  83  Linux
/dev/hda11          418       629   1602688+  83  Linux
```

A few unused partitions exist. These are for installing a second Linux distribution or as expansion space for growing directory trees.

Mount file systems

Mounting the above file systems properly is accomplished with the following `/etc/fstab`:

```
# /etc/fstab: static file system information.
#
# file system mount point type options        dump pass
/dev/hda3 / ext2 defaults,errors=remount-ro 0 1
/dev/hda5 none swap sw                      0 0
proc /proc proc defaults 0 0
/dev/fd0 /floppy auto defaults,user,noauto 0 0
/dev/cdrom /cdrom iso9660 defaults,ro,user,noauto 0 0
#
```

```
# keep partition separate
/dev/hda7 /home ext2 rw 0 2
/dev/hda8 /var ext2 rw 0 2
/dev/hda6 /var/lib/cvs ext2 rw 0 2
/dev/hda9 /usr ext2 rw 0 2
/dev/hda10 /var/cache/apt/archives ext2 rw 0 2

# very big partition for proxy cache
/dev/hda11 /var/spool/squid ext2 rw 0 2

# backup bootable DOS
/dev/hda1 /mnt/dos vfat rw,noauto 0 0
# backup bootable Linux system (not done)
/dev/hda2 /mnt/linux ext2 rw,noauto 0 0
#
# nfs mounts
mickey:/ /mnt/mickey nfs ro,noauto,intr 0 0
goofy:/ /mnt/goofy nfs ro,noauto,intr 0 0
# minnie:/ /mnt/minnie smbfs ro,soft,intr,credentials={filename} 0 2
```

For NFS, I use `noauto,intr` combined with the default `hard` option. This way, it is possible to recover from a hung process due to a dead connection using Control-C.

For a Windows machine connected with Samba (smbfs), `rw,auto,soft,intr` may be good idea. See 'Samba configuration' on page 37.

For a floppy drive, using `noauto,rw,sync,user,exec` instead prevents file corruption after accidental disk eject before unmount, but this slows the write process.

Autofs mount

Key points to auto mount:

- Load the `vfat` module to allow `/etc/auto.misc` to contain `-fstype=auto`:

```
# modprobe vfat # prior to the floppy access attempt
 ... or to automate this settings,
# cat >>/etc/modules
vfat
^D
 ... and reboot the system.
```

- Set `/etc/auto.misc` as follows:

```
floppy -fstype=auto,sync,nodev,nosuid,gid=100,umask=000 :/dev/fd0
 ... where gid=100 is "users".
```

- Create links in `/home/user`, `cdrom` and `floppy`, that point to `/var/autofs/misc/cdrom` and `/var/autofs/misc/floppy` respectively.
- Make *user* as a member of “users” group.

NFS mount

The external Linux NFS server (goofy) resides behind a firewall (gateway). I have a very relaxed security policy on my LAN since I am the only user. To enable NFS access, the NFS server side needs to add `/etc/exports` as follows:

```
# /etc/exports: the access control list for file systems which may be
#               exported to NFS clients.  See exports(5).
/       (rw,no_root_squash)
```

This is needed to activate the NFS server in addition to installing and activating the NFS server and client.

For simplicity, I usually create a single partition of 2GB for an experimental or secondary lazy Linux install. I optionally share swap and `/tmp` partitions for these installs. A multi-partition scheme is too involved for these usages. If only a simple console system is needed, 500MB may be more than sufficient.

3.1.11 DRAM memory guidelines

Following are rough guidelines for DRAM.

```
  4 MB:  Bare minimum for Linux kernel to function.
 16 MB:  Minimum for reasonable console system.
 32 MB:  Minimum for simple X system.
 64 MB:  Minimum for X system with GNOME/KDE.
128 MB:  Comfortable for X system with GNOME/KDE.
256+MB:  Why not if you can afford it?  DRAM is cheap.
```

Using the boot option `mem=4m` (or lilo `append="mem=4m"`) will show how the system would perform with 4MB of memory installed. A lilo boot parameter is needed for a system containing more than 64MB of memory with an old BIOS.

3.1.12 Swap space

I use the following guidelines for swap space:

- Each swap partition is < 128 MB (if old 2.0 kernel), < 2 GB (in recent kernels)
- Total = either (1 to 2 times installed RAM) or (128 MB to 2 GB) as a guideline
- Spread them on different drives and mount all of them with `sw,pri=1` options in `/etc/fstab`. This ensures that the kernel does a striping RAID of the swap partitions and offers the maximum swap performance.
- Use a central portion of the hard disk when possible.

Even if you never need it, some swap space (128MB) is desirable so the system will slow down before it crashes hard with a program which leaks memory.

3.2 Bash configuration

I modify shell start-up scripts to my taste across the system:

```
/etc/bash.bashrc        Replace with private one
/etc/profile            Keep distribution copy ( \w -> \W)
/etc/skel/.bashrc       Replace with private copy
/etc/skel/.profile      Replace with private copy
/etc/skel/.bash_profile Replace with private copy
~/.bashrc               Replace with private copy for all accounts
~/.profile              Replace with private copy for all accounts
~/.bash_profile         Replace with private copy for all accounts
```

See details in my example scripts (`examples/`). I like a transparent system, so I set `umask` to 002 or 022.

`PATH` is set by the following configuration files in this order:

```
/etc/login.defs  - before the shell sets PATH
/etc/profile     (may call /etc/bash.bashrc)
~/.bash_profile  (may call ~/.bashrc)
```

3.3 Mouse configuration

3.3.1 PS2 mice

In the case of a PS/2-connector mouse on an ATX motherboard, the signal flow should be:

```
mouse -> /dev/psaux -> gpm -> /dev/gpmdata = /dev/mouse -> X
```

This allows the keyboard and mouse to be unplugged and reinitialized by restarting gpm upon reconnect. X will stay alive!

For a Logitech 3-button **PS2** mice, configuration combinations should be:

```
/etc/gpm.conf               /etc/X11/X86Config or X86Config-4
==============================================================
device=/dev/psaux              Section "Pointer"
responsiveness=                  Protocol     "IntelliMouse"
repeat_type=ms3                  Device       "/dev/gpmdata"
type=ps2auto   (Woody)
append=""
---------------------------------------------------------------
device=/dev/psaux              Section "Pointer"
responsiveness=                  Protocol     "IntelliMouse"
repeat_type=raw                  Device       "/dev/gpmdata"
type=ps2auto  (Woody)
append=""
```

If a normal 2-button PS2 mouse is used, set the X protocol to `Microsoft` and enable `Emulate3Buttons`. For a scroll mouse, you can adjust X to the real protocol, such as `IMPS/2`. Create a symlink `/dev/gpmdata` –> `/dev/mouse` to make some configuration utilities happy. See my example scripts for details (`examples/`).

For some recent thin Toshiba notebook PCs: Activate `gpm` before PCMCIA in the System-V init script. This keeps `gpm` from crashing. Weird but true.

3.3.2 USB mice

Make sure you have:

- "Input Core Support" and "Input Core Support/Mouse Support" enabled in the kernel or as modules.
- "Support for USB", "Preliminary USB device filesystem", "UHCI or OHCI", and "USB HID Support" enabled in the kernel or as modules.

If you're not using devfs, create a device node `/dev/input/mice` with major 13 and minor 63 as follows:

```
# cd /dev
# mkdir input
# mknod input/mice c 13 63
```

For Logitech 3-button **USB** mice, configuration combinations should be:

```
/etc/gpm.conf                  /etc/X11/X86Config or X86Config-4
=================================================================
device=/dev/input/mice         Section "InputDevice"
responsiveness=                  Option "Protocol" "ImPS/2"
repeat_type=ms3                  Option "Device" "/dev/input/mice"
type=ps2                         Option "ZAxisMapping" "4 5"
append=""                      EndSection
```

[This USB mouse section was written by Jan Michael C Alonzo <`jmalonzo@softhome.net`>.]

See Linux USB Project (`http://www.linux-usb.org/`) for more information.

3.4 NFS configuration

Set up NFS by setting `/etc/exports`.

```
# echo "/ *.domainname-for-lan-hosts(rw,no_root_squash,nohide)" \
        >> /etc/exports
```

See my example scripts for details (`examples/`).

3.5 Samba configuration

References:

- `http://www.samba.org/`
- `samba-doc` package

Setting up Samba with "share" mode is much easier since this creates WfW-type share drives. But it is preferable to set it up with "user" mode.

Samba can be configured through `debconf` or `vi`:

```
# dpkg-reconfigure --priority=low samba # in Woody
# vi /etc/samba/smb.conf
```

See my example scripts for details (`examples/`).

Adding a new user to the smbpasswd file can be done via `smbpasswd`:

```
$su -c "smbpasswd -a username"
```

Make sure to use encrypted passwords for optimum compatibility.

Set `os level` according to the following system equivalences (the larger the number, the higher the priority as server):

```
0:      Samba with a loose attitude (will never become a master browser)
1:      Wfw 3.1, Win95, Win98, Win/me?
16:     Win NT WS 3.51
17:     Win NT WS 4.0
32:     Win NT SVR 3.51
33:     Win NT SVR 4.0
255:    Samba with mighty power
```

Make sure that users are members of the group owning the directory that gives shared access and that the directory path has its execution bit set to access.

3.6 Printer configuration

The traditional method is `lpr/lpd`. There is a new CUPS™ system (Common UNIX Printing System). PDQ is another approach. See the Linux Printing HOWTO (`http://www.tldp.org/HOWTO/Printing-HOWTO.html`) for more information.

3.6.1 `lpr/lpd`

For the `lpr/lpd` type spoolers (`lpr`, `lprng`, and `gnulpr`), set up `/etc/printcap` as follows if they are connected to a PostScript or text-only printer (the basics):

```
lp|alias:\
        :sd=/var/spool/lpd/lp:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp0:
```

Meaning of the above lines:

- Head line: *lp* — name of spool, *alias* = alias
- mx#0 — max file size unlimited
- sh — suppress printing of burst page header

- lp=/dev/lp0 — local printer device, or port@host for remote

This is a good configuration if you are connected to a PostScript printer. Also, when printing from a Windows machine through Samba, this is a good configuration for any Windows-supported printer (no bidirectional communication is supported). You have to select the corresponding printer configuration on the Windows machine.

If you do not have a PostScript printer, you need to set up a filtering system using `gs`. There are many auto-configuration tools provided for setting up `/etc/printcap`. Any of these combinations is an option:

- `gnulpr`, (`lpr-ppd`) and `printtool` — I use this.
- `lpr` and `apsfilter`
- `lpr` and `magicfilter`
- `lprng` and `lprngtool`
- `lprng` and `apsfilter`
- `lprng` and `magicfilter`

In order to run GUI configuration tools such as `printtool`, see 'Gain root in X' on page 110 to gain root privilege. Printer spools created with `printtool` use `gs` and act like PostScript printers. So when accessing them, use PostScript printer drivers. On the Windows side, "Apple LaserWriter" is the standard one.

3.6.2 CUPS™

Install the Common UNIX Printing System (or CUPS™):

```
# apt-get install cupsys cupsomatic-ppd
# apt-get install cupsys-bsd cupsys-driver-gimpprint
```

Then configure the system using any Web browser:

```
$ mybrowser http://localhost:631
```

3.7 Other host installation hints

3.7.1 Install a few more packages after initial install

Once you have made it this far, you have a small but functioning Debian system. It is a good time to install bigger packages.

- Run `tasksel`. See 'Install *task* with `tasksel`' on page 58.

  You may choose these if you need them:
  - End-user — X window system
  - Development — C and C++
  - Development — Python
  - Development — Tcl/Tk
  - Miscellaneous — TeX/LaTeX environment
  - For others — I prefer to use `tasksel` as a guide by looking into their components listed under <Task Info> and installing them selectively through `dselect`.

- Run `dselect`.

  Here the first thing you may want to do is select your favorite editor and any programs you need. You can install many Emacs variants at the same time. See '`dselect` – global configuration' on page 66 and 'Popular editors' on page 129.

  Also you may replace some of the default packages with full-featured ones.
  - lynx-ssh (instead of lynx)
  - ...

- ...

I usually edit `/etc/inittab` for easy shutdown.

```
...
# What to do when CTRL-ALT-DEL is pressed.
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -h now
...
```

3.7.2 Modules

Modules for the device drivers are configured during the initial installation. `modconf` provides menu-driven module configuration afterward. This program is quite useful when some modules were left out during the initial installation or a new kernel was installed after the initial installation.

All preloading module names need to be listed in `/etc/modules`. I also use `lsmod` and `depmod` to control them manually.

Also make sure to add a few lines in `/etc/modules` to handle ip-masquerading (ftp, etc.) for 2.4 kernels. See 'The modularized 2.4 kernel' on page 75, specifically 'Network function' on page 76.

3.7.3 CD-RW basic setup

Edit the following files:

```
/etc/lilo.conf  (add append="hdc=ide-scsi ignore=hdc",
                 run lilo to activate)
/dev/cdrom      (symlink # cd /dev; ln -sf scd0 cdrom)
/etc/modules    (add "ide-scsi" and "sg". If needed "sr" after this.)
```

See 'CD-writer' on page 102 for details.

3.7.4 Large memory and auto power-off

Edit `/etc/lilo.conf` as follows to set boot-prompt parameters for large memory (for 2.2 kernels) and auto power-off (for apm):

```
append="mem=128M apm=on apm=power-off"
```

Run `lilo` to install these settings. `apm=power-off` is needed for an SMP-kernel. The same can be done directly by entering options at the boot prompt. See 'Other boot tricks with the boot prompt' on page 81.

If apm is compiled as a module, as in Debian default 2.4 kernels, run `# insmod apm power_off=1` after boot or set `/etc/modules` by:

```
# echo "apm power_off=1" >>/etc/modules
```

Alternatively, compiling ACPI support achieves the same goal with newer kernels and seems to be more SMP-friendly (this requires a newer motherboard). The 2.4 kernel on newer motherboards should detect large memory correctly.

```
CONFIG_PM=y
CONFIG_ACPI=y
...
CONFIG_ACPI_BUSMGR=m
CONFIG_ACPI_SYS=m
```

and add the following lines in `/etc/modules` in this order:

```
ospm_busmgr
ospm_system
```

Or recompile the kernel with all of the kernel options above set to “y”. In any case, none of the boot-prompt parameters are needed with ACPI.

3.7.5 Strange access problems with some websites

Recent Linux kernels enable ECN by default, which may cause access problems with some websites on bad routers. To check ECN status:

```
# cat /proc/sys/net/ipv4/tcp_ecn
 ... or
# sysctl net.ipv4.tcp_ecn
```

To turn it off, use:

```
# echo "0" > /proc/sys/net/ipv4/tcp_ecn
 ... or
# sysctl -w net.ipv4.tcp_ecn=0
```

To disable TCP ECN on every boot, edit `/etc/sysctl.conf` and add:

```
net.ipv4.tcp_ecn = 0
```

3.7.6 Dial-up PPP configuration

Install the `pppconfig` package to set up dial-up PPP access.

```
# apt-get install pppconfig
# pppconfig
 ... follow the directions to configure dial-up PPP
# adduser user_name dip
 ... allow user_name to access dial-up PPP
```

Dial-up PPP access can be initiated by the user (*user_name*):

```
$ pon ISP_name   # start PPP access to your ISP
 ... enjoy the Internet
$ poff ISP_name # stop PPP access, ISP_name optional
```

See `/usr/share/doc/ppp/README.Debian.gz` for more details.

Alternatively, the `wvdial` package may be used to set up dial-up PPP access.

3.7.7 Other configuration files to tweak in `/etc`

You may want to add an `/etc/cron.deny` file, missing from the standard Debian install (you can copy `/etc/at.deny`).

Chapter 4

Debian tutorials

This section provides a basic orientation to the Linux world for the real newbie. If you have been using Linux for a while, use it as a reality check.

4.1 Information sources

Look into the Debian Documentation Project (DDP) (`http://www.debian.org/doc/`), which has the most authoritative references for Debian. Many of these documents are usually installed in `/usr/share/doc/`. Also look into `/usr/share/doc-base/`, which provides pointers to the documents on the system. Add `export CDPATH=.:/usr/share/doc:/usr/src/local` to `~/.bash_profile` for easier access to documentation directories.

The Linux Documentation Project (LDP) (`http://www.tldp.org/`) has the most authoritative general Linux references. LDP contents are usually installed in `/usr/share/doc/HOWTO/`.

Navigate through documents on local and remote FTP sites using `F9` in Midnight Commander (see 'Midnight Commander (MC)' on page 48).

4.2 The Linux console

4.2.1 Login

In an ordinary Linux system, there are 6 independent pseudo-terminals. Switch from one to another by pressing the `Left-Alt` key and `F1` - `F6` keys simultaneously. Each pseudo-terminal allows independent login to accounts. The multi-user environment is a great Unix feature, and very addictive.

It is considered a good Unix habit to login to a regular user account for most purposes. I have to admit I used to use the superuser account (root) more than needed just because of its ease and my sloppiness.

Now I usually use a regular account with the commands `sudo`, `super` or `su -c` to gain limited root access.

4.2.2 Add a user account

After system installation, I usually add a regular user account. If the username is "penguin",

```
# adduser penguin
```

will create it.

I use the `vigr` command to edit `/etc/group` as follows:

```
src:x:40:admin, debian, ...
staff:x:50:admin
...
```

I use the `staff` group for users who do administrative duties and have the exclusive `su` privilege (see "'Why GNU `su` does not support the `wheel` group"' on page 99) and `src` for CVS (see 'CVS' on page 137).

In the default install system, the `staff` group owns `/home`, making its members suitable for maintaining user accounts, while the `src` group owns `/usr/src`, used for kernel compile, etc.

Check out `adduser`, `addgroup`, `vipw`, `vipw -s`, `vigr`, and `vigr -s` for configuring users and groups properly.

4.2.3 How to shut down

Just like any modern OS where files are cached in memory, Linux needs a proper shutdown procedure before power can safely be turned off. Here is the command in multi-user mode:

```
# shutdown -h now
```

Here is the command in single-user mode:

```
# poweroff -i -f
```

Wait until the system displays “System halted” then shut off power. If apm has been turned on by the BIOS and Linux, the system will power down by itself. See ‘Large memory and auto power-off’ on page 41 for details.

4.2.4 Command-line editing

The default shell, `bash`, has history-editing capability. Just use the up-arrow key to enter the history and then use cursor keys as you would expect. Other important keystrokes to remember:

```
Ctrl-C:                    Terminate program
Ctrl-D:                    Terminate input
Ctrl-S:                    Halt output to screen
Ctrl-Q:                    Reactivate output to screen
Ctrl-Alt-Del:              Reboot/halt system (see /etc/inittab)
Lt-click-and-drag-mouse: Select and copy to the clipboard (gpm)
Ctrl-click-mouse:          Paste the clipboard to the cursor (gpm)
```

On a normal Linux console, only the left-hand `Ctrl` and `Alt` keys work as expected.

4.2.5 Very basic commands to remember

The following are very basic Unix commands:

```
ls, ls -al, ls -d, pwd, cd, cd ~user, cd -,
cat /etc/passwd, less, bg, fg, kill, killall,
uname -a, type commandname, sync, netstat,
ping, traceroute, top, vi, ps aux, tar, zcat,
grep, ifconfig, ...
```

Check their meaning by entering the commands at a command prompt or by entering `man` or `info` plus the command name. Many Linux commands will display brief help information if you invoke them in one of the following ways:

```
$ commandname --help
$ commandname -h
```

`whatis` *`commandname`* gives a one-line summary of any command on the system for which there is a manual entry.

4.2.6 X Window System

To start the X Window System from the console:

```
# exec startx
```

Right-clicking the root window will bring up menu selections.

4.2.7 Important keyboard commands

Some important keystrokes to remember for the Linux console:

```
Alt-F1 thru F6:       Switch to other pseudo-terminals
Ctrl-Alt-F1 thru F6:  Switch to other pseudo-terminals
                      (from an X-window, DOSEMU, etc.)
Alt-F7:               Switch back to X-window
Ctrl-Alt-minus:       Change screen resolution in X-window
Ctrl-Alt-plus:        Change screen resolution opposite way in X-window
Ctrl-Alt-Backspace:   Terminate X-windows
Alt-X, Alt-C, Alt-V:  Usual Windows/Mac Cut, Copy, Paste key
        combinations with Ctrl- keys are replaced by these Alt- keys
        in some programs such as Netscape Composer.
```

4.3 Midnight Commander (MC)

Midnight Commander (MC) is a GNU “Swiss army knife” for the Linux console and other terminal environments.

4.3.1 Install MC

```
# apt-get install mc
```

Then add the following function to `~/.bashrc` (or to `/etc/bash.bashrc`, called from `.bashrc`).

```
mc ()
{
  mkdir -p ~/.mc/tmp 2> /dev/null
```

```
      chmod 700 ~/.mc/tmp
      MC=~/.mc/tmp/mc-$$
      /usr/bin/mc -P "$@" > "$MC"
      cd "$(cat $MC)"
      rm -f "$MC"
      unset MC;
  }
```

This enables MC to change working directory upon exit.

If one is in a terminal, like `kon` and `Kterm` for Japanese, which utilizes certain graphics characters, adding `-a` to MC's command line may help prevent problems.

4.3.2 Start MC

```
$ mc
```

MC takes care of all file operations through its menu, requiring minimal user effort.

4.3.3 File manager

The default is 2 directory panels containing file lists. Another useful mode is to set the right window to "information" to see file access privilege information, etc. Following are some essential keystrokes. With the `gpm` daemon running, one can use a mouse, too. (Make sure to press the shift key to obtain the normal behavior of cut and paste in MC.)

- `F1`: Help menu
- `F3`: Internal file viewer
- `F4`: Internal editor
- `F9`: Activate pulldown menu
- `F10`: Exit Midnight Commander
- `Tab`: Move between 2 windows
- `Insert`: Mark file for a multiple-file operation such as copy
- `Del`: Delete file (Be careful—set MC to safe delete mode.)
- Cursor keys: Self-explanatory

4.3.4 Command-line tricks

- Any `cd` command will change the directory shown on the selected screen.

- `Control-Enter` or `Alt-Enter` will copy a filename to the command line. Use this with the `cp` or `mv` command together with command-line editing.

- `Alt-Tab` will show shell filename expansion choices.
- One can specify the starting directory for both windows as arguments to MC; for example, `mc /etc /root`.
- `Esc` + *numberkey* == `Fn` (i.e., `Esc` + `1` = `F1`, etc.; `Esc` + `0` = `F10`)
- `Esc` key == `Alt` key (= `Meta`, `M-`); i.e., type `Esc` + `c` for `Alt-c`

4.3.5 Editor

The internal editor has an interesting cut-and-paste scheme. Pressing `F3` marks the start of a selection, a second `F3` marks the end of selection and highlights the selection. Then you can move your cursor. If you press `F6`, the selected area will be moved to the cursor location. If you press `F5`, the selected area will be copied and inserted at the cursor location. `F2` will save the file. `F10` will get you out. Most cursor keys work intuitively.

This editor can be directly started on a file:

```
$ mc -e filename_to_edit
$ mcedit filename_to_edit
```

This is not a multi-window editor, but one can use multiple Linux consoles to achieve the same effect. To copy between windows, use `Alt-Fn` keys to switch virtual consoles and use "File->Insert file" or "File->Copy to file" to move a portion of a file to another file.

This internal editor can be replaced with any external editor of choice.

Also, many programs use environment variables `EDITOR` or `VISUAL` to decide which editor to use. If you are unconfortable with vim, set these to `mcedit` by adding these lines to `~/.bashrc`:

```
...
export EDITOR=mcedit
export VISUAL=mcedit
...
```

I do recommend setting these to `vim` if possible. Getting used to vi(m) commands is the right thing to do, since they are always there in the Linux/Unix world.

4.3.6 Viewer

Very smart viewer. This is a great tool for searching words in documents. I always use this for files in the `/usr/share/doc` directory. This is the fastest way to browse through masses of Linux information. This viewer can be directly started like so:

```
$ mc -v filename_to_view
```

(Note that some packages violate policy and still store their documents under `/usr/doc`.)

4.3.7 Auto-start features

Press `Enter` on a file, and the appropriate program will handle the content of the file. This is a very convenient MC feature.

```
executable:         Execute command
man, html file:     Pipe content to viewer software
tar, gz, rpm file: Browse its contents as if subdirectory
```

In order to allow these viewer/virtual file features to function, viewable files should not be set as executable. Change their status using the `chmod` command or via the MC file menu.

4.3.8 FTP virtual file system

MC can be used to access files over the Internet using FTP. Go to the menu by pressing `F9`, then type `p` to activate the FTP virtual file system. Enter a URL in the form `username:passwd@hostname.domainna` which will retrieve a remote directory that appears like a local one.

4.4 Further study

There are many good Unix entry-level references out there. O'Reilly's books are usually safe bets for good guidebooks in any field of computer topics. The LDP document Tips-HOWTO (`http://www.tldp.org/HOWTO/Tips-HOWTO.html`) is another resource to check. See 'Support for Debian' on page 165 for more resources.

Chapter 5

Upgrading a distribution

Official release notes for upgrading are located at `http://www.debian.org/releases/stable/releasenotes` and `http://www.debian.org/releases/testing/releasenotes` (work in progress).

5.1 Transition preparation ("stable" to "testing")

Network upgrade to "testing" can be done as follows (run the script go-woody (`examples/`) to do this in one command):

```
# cd /etc/apt
# cp -f sources.list sources.old
# :>sources.list
# cd /
# apt-setup noprobe
 ... select http or ftp
# cd /etc/apt
# grep -e "^deb " sources.list             >sources.deb
# grep -e "^deb-" sources.list             >sources.src
# sed -e "s/^d/#d/" \
  /usr/share/doc/apt/examples/sources.list >sources.list
# sed -e "s/stable/testing/"  \
  sources.deb >>sources.list
# apt-get update
# apt-get install apt apt-utils
# cat >preferences <<EOF
 Package: *
```

```
 Pin: release a=testing
 Pin-Priority: 700

 Package: *
 Pin: release a=unstable
 Pin-Priority: 70

 EOF
# sed -e "s/stable/unstable/" sources.deb \
  >>sources.list
# sed -e "s/stable/unstable/" sources.src | \
 sed -e "s/^deb-/#deb-/"  >>sources.list
```

A guideline for `/etc/apt/preferences` (see `apt_preferences(5)`):

```
track stable:            change  Pin-Priority of testing  to  80
track testing:           keep as is (install unstable by /unstable)
track testing(unstable): change  Pin-Priority of unstable to 600
track unstable(testing): change  Pin-Priority of unstable to 800
```

A guideline for the choice of Pin-Priority is to move from the top to bottom in the above table as the time moves from a time immediately after a distribution release to a time of freeze for the next release.

Examples of `/etc/apt/preferences` which lock some key packages to the more mature version while tracking the less mature version for other nonessential packages are available in the examples subdirectory (`examples/`) as `preferences.testing` and `preferences.unstable`. On the other hand, `preferences.stable` forces all packages to be downgraded to “stable”.

Make sure to set up APT to use a proxy, if necessary, by setting the `http_proxy` environment variable or set the http value in `/etc/apt/apt.conf`.

The procedure described in this section only upgrades APT and a minimum set of packages to avoid dependency-related problems.

5.2 Upgrade to “testing”

After the above preparation, the system can be upgraded.

5.2.1 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 :)

5.2.2 Deprecated upgrade practice using `apt-get`

The use of `apt-get` described below is widespread but it is *not* recommended for system upgrades. If you need to upgrade without `dselect` after Woody, consider `aptitudes` and other options.

If a system does not have many packages or the Debian archive did not have major changes, the following may be sufficient (sometimes).

```
# apt-get update # always do this before upgrade
 ... to upgrade the system with "depends" selections:
# apt-get upgrade                     # always do this before upgrade
 ... to upgrade the whole system with "depends" selections:
# apt-get -u dist-upgrade
 ... or to upgrade and stay with current dselect settings (new, better):
# apt-get -u dselect-upgrade         # use dselect setup result
```

Since this upgrade method uses `apt-get`, its handling of *recommends* and *suggests* is limited. See 'Package dependencies' on page 15.

5.3 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)`.

5.4 Optimized `sources.list`

Create `sources.list` automatically, based on latency and bandwidth.

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

`netselect-apt` is very similar to `apt-spy`. It creates a more complete `sources.list`, but uses an inferior method of choosing the best mirror (ping time comparison). `apt-setup` is the manual method for selecting the mirrors in `sources.list`, but is still the best way to choose mirrors until `apt-spy` improves.

These optimization efforts did not produce significant improvement for me. Just using nearby sites chosen by `apt-setup` was sufficient.

Chapter 6

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`).

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 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. See 'Package dependencies' on page 15.

6.1.2 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
...
```

6.2 Debian survival commands

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

Also refer to 'Debian System installation hints' on page 27, 'Upgrading a distribution' on page 53 and 'Rescue editors' on page 129.

6.2.1 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
```

6.2.2 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.

6.2.3 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`.

6.2.4 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
```

More support resources are listed at ‘Support for Debian’ on page 165.

6.2.5 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` and 'The `equivs` package' on page 70.

6.2.6 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. (See 'The `pool` directory' on page 9.)

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
```

6.2.7 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.

6.2.8 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.)

6.2.9 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.

6.3 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**. :-)

6.3.1 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
```

6.3.2 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}
```

6.3.3 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
...
```

6.3.4 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 ...
```

6.3.5 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$"
```

6.3.6 `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.

6.3.7 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
```

6.3.8 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
```

6.3.9 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
```

6.3.10 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
```

6.3.11 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.

6.3.12 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).

6.4 Other Debian peculiarities

6.4.1 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 (see 'Preservation of the local configuration' on page 13).

```
# 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.

6.4.2 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.
```

6.4.3 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`.

6.4.4 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.

6.4.5 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` as in 'Access control through PAM and login' on page 98. The `pidof` command is also useful (see `pidof(8)`).

X11 doesn't allow TCP/IP (remote) connections by default in recent versions of Debian. See 'TCP/IP connection to X' on page 108. X forwarding in SSH is also disabled. See 'Remote X connection: `ssh`' on page 109.

Chapter 7

The Linux kernel under Debian

Debian has its own method of recompiling the kernel and related modules. See also 'Debian and the kernel' on page 24.

7.1 Kernel recompile

The use of `gcc`, `binutils` and `modutils` from Debian `unstable` may help when compiling the latest Linux kernel.

7.1.1 Debian standard method

Use the new `kernel-package` in Woody. Also watch out for bug reports on `gcc`, `binutils` and `modutils`.

Compiling a custom kernel from source under a Debian system requires special care. Use the new `--append_to_version` with `make-kpkg` to build multiple kernel-images.

```
# apt-get install debhelper modutils kernel-package libncurses5-dev
# apt-get install kernel-source-2.4.18   # use latest version
# vi /etc/kernel-pkg.conf                # input my name and email
$ cd /usr/src                            # build directory
$ tar --bzip2 -xvf kernel-source-2.4.18.tar.bz2
$ cd kernel-source-2.4.18                # if this is your kernel source
$ rm -rf */pcmcia
    # [OPTIONAL] if one wants to use modules from pcmcia-cs or no pcmcia
$ cp /boot/config-2.4.18-386 .config     # get current config as default
```

```
$ make menuconfig                              # customize as one wishes
$ make-kpkg clean                              # must run (per: man make-kpkg)
$ fakeroot make-kpkg --append_to_version -486 --initrd \
        --revision=rev.01 kernel_image \
        modules_image # modules_image is for pcmcia-cs* etc.
$ cd ..
# dpkg -i kernel-image*.deb pcmcia-cs*.deb # install
```

`make-kpkg kernel_image` actually does `make oldconfig` and `make dep`. Do not use `--initrd` if initrd is not used. The current Debian `initrd` boot script requires a `cramfs` kernel patch, if you obtain kernel source from non-Debian archives; see `http://bugs.debian.org/149236`.

One can avoid `rm -fr */pcmcia` by selecting "General setup —>" to "PCMCIA/CardBus support —>" in `make menuconfig` and setting the configuration to "< > PCMCIA/CardBus support" (i.e., uncheck the box).

On an SMP machine, set CONCURRENCY_LEVEL according to `kernel-pkg.conf(5)`.

7.1.2 Classic method

Get pristine sources from:
- Linux: `http://www.kernel.org/`
- pcmcia-cs: `http://pcmcia-cs.sourceforge.net/`

or use equivalent source in Debian and do the following:

```
# cd /usr/src
# tar xfvz linux-whatever.tar.gz
# rm -rf linux
# ln -s linux-whatever linux
# tar xfvz pcmcia-cs-whatever.tar.gz
# ln -s pcmcia-cs-whatever pcmcia
# cd linux
# rm -rf */pcmcia
    # [OPTIONAL] if one wants to use modules from pcmcia-cs or no pcmcia
# make menuconfig
... configure stuff ...
# make dep
# make bzImage
... edits for lilo / grub ...
... move /usr/src/linux/arch/i386/boot/bzImage to boot ...
... /sbin/lilo or whatever you do for grub
# make modules; make modules_install
```

```
# cd ../pcmcia
# make config
# make all
# make install
... add needed module names to /etc/modules
# shutdown -r now
... boot to new kernel ...
```

7.2 The modularized 2.4 kernel

The new Debian 2.4 kernels provided by kernel-image-2.4.*NN* are very modularized. You have to make sure those modules are activated to make the kernel function as you intend.

7.2.1 PCMCIA

`/etc/modules` needs to contain the following for PCMCIA to function:

```
# ISA PnP driver
isa-pnp
# Low level PCMCIA driver
# yenta_socket # does not seem to be needed in my case
```

The rest is taken care of by PCMCIA scripts (from the `pcmcia-cs` package), `depmod` and `kmod`. I think I needed `isa-pnp` because my laptop is an old ISA-PCMCIA. Recent laptops with CardBus/PCMCIA may not require this.

Voice of the generous Miquel van Smoorenburg `<miquels@cistron.nl>`:

"I simply removed the entire pcmcia stuff from the laptop here at work, including the cardmgr etc and just installed a 2.4 kernel with cardbus support, and the new `hotplug` package from woody.

As long as you only have 32-bit cards you don't need the pcmcia package; 2.4 has cardservices built in. And the standard tulip driver should work fine with your dlink card.

—Mike."

7.2.2 SCSI

[NOT TESTED] `/etc/modules` needs to contain the following for SCSI to function:

```
# SCSI core
scsi_mod
# SCSI generic driver
sg
# SCSI disk
sd_mod
# All other needed HW modules
...
```

`depmod` may take care of some of the above modules.

7.2.3 Network function

`/etc/modules` needs to contain the following for extra network function:

```
# net/ipv-4
ip_gre
ipip

# net/ipv-4/netfilter
# iptable (in order)
ip_tables
ip_conntrack
ip_conntrack_ftp
iptable_nat
iptable_filter
iptable_mangle
#
ip_nat_ftp
ip_queue
#
ipt_LOG
ipt_MARK
ipt_MASQUERADE
ipt_MIRROR
ipt_REDIRECT
ipt_REJECT
ipt_TCPMSS
ipt_TOS
ipt_limit
ipt_mac
```

```
ipt_mark
ipt_multiport
ipt_owner
ipt_state
ipt_tcpmss
ipt_tos
ipt_unclean
#
#ipchains
#ipfwadm
```

The preceding may not be optimized. `depmod` may take care of some of the above modules.

7.2.4 EXT3 filesystem (> 2.4.17)

Enabling a journaling filesystem with the EXT3 FS involves the following steps using a Debian precompiled kernel-image (> 2.4.17) package:

```
# cd /etc; mv fstab fstab.old
# sed 's/ext2/ext3,ext2/g' <fstab.old >fstab
# vi /etc/fstab
... set root filesystem type to "auto" instead of "ext3,ext2"
# cd /etc/mkinitrd
# echo jbd >>modules
# echo ext3 >>modules
# echo ext2 >>modules
# cd /
# apt-get update; apt-get install kernel-image-2.4.17-686-smp
... install latest kernel and set up boot (lilo is run here)
# tune2fs -j -i 0 /dev/hda1
# tune2fs -j -i 0 /dev/hda2
... For all EXT2 FS's converted to EXT3
# shutdown -r now
```

Now EXT3 journaling is enabled. Using `ext3,ext2` as the `fstab` “type” entry ensures safe fallback to EXT2 if the kernel does not support EXT3 for non-root partitions.

If you have previously installed a 2.4 kernel and do not wish to reinstall, perform the above steps up to the `apt-get` commands, then:

```
# mkinitrd -o /boot/initrd.img-2.4.17-686-smp /lib/modules/2.4.17-686-smp
```

```
# lilo
# tune2fs -j -i 0 /dev/hda1
# tune2fs -j -i 0 /dev/hda2
... for all EXT2 FS's converted to EXT3
# shutdown -r now
```

Now EXT3 journaling is enabled.

If `/etc/mkinitrd/modules` was not set when `mkinitrd` was run and you would like to add some modules at boot time:

```
... at initrd prompt to gain shell (5 sec.), type RETURN
# insmod jbd
# insmod ext3 # modprobe ext3 may take care of everything
# insmod ext2
# ^D
... continue booting
```

At the system boot screen (`dmesg`), "cramfs: wrong magic" appears but this is known to be harmless. This will be resolved in a later release (2002/8). See `http://bugs.debian.org/135537` and the EXT3 File System mini-HOWTO (`http://www.symonds.net/~rajesh/howto/ext3/index.html`) or `/usr/share/doc/HOWTO/en-txt/mini/extra/ext3-mini-HOWTO.gz` for more information.

Some systems are reported to experience severe kernel lock-up if EXT3 is enabled but I had no problem (as of 2.4.17).

7.2.5 Realtek RTL-8139 support in 2.4

For whatever reason, the RTL-8139 support module is no longer called rtl8139, it's now called 8139too. Just edit your `/etc/modules` to reflect this change when upgrading a 2.2 kernel to a 2.4 kernel.

Chapter 8

Debian tips

8.1 Booting the system

See the LDP BootPrompt-HOWTO (`http://www.tldp.org/HOWTO/BootPrompt-HOWTO.html`) for detailed information on the boot prompt.

8.1.1 “I forgot the root password!” (1)

It is possible to boot a system and log on to the root account without knowing the root password as long as one has access to the console keyboard. (This assumes there are no password requests from the BIOS or from a boot-loader such as `lilo` that would prevent one from booting the system.)

This is a procedure which requires no external boot disks and no change in BIOS boot settings. Here, “Linux” is the label for booting the Linux kernel in the default Debian install.

At the `lilo` boot screen, as soon as `boot:` appears (you must press a shift key at this point on some systems to prevent automatic booting), enter:

```
boot: Linux init=/bin/sh
```

This causes the system to boot the kernel and run `/bin/sh` instead of its standard `init`. Now you have gained root privileges and a root shell. Since `/` is currently mounted read-only and many disk partitions have not been mounted yet, you must do the following to have a reasonably functioning system.

```
init-2.03# mount -n -o remount,rw /
init-2.03# mount -avt nonfs,noproc,nosmbfs
```

```
init-2.03# cd /etc
init-2.03# vi passwd
init-2.03# vi shadow
```

(If the second data field in `/etc/passwd` is "x" for every username, your system uses shadow passwords, and you must edit `/etc/shadow`.) To disable the root password, edit the second data field in the password file so that it is empty. Now the system can be rebooted and you can log on as root without a password. When booting into runlevel 1, Debian (at least after Potato) requires a password, which some older distributions did not.

It is a good idea to have a minimum editor in `/bin` in case `/usr` is not accessible (see 'Rescue editors' on page 129).

Also consider installing the `sash` package. When the system becomes unbootable, execute:

```
boot: Linux init=/bin/sash
```

`sash` serves as an interactive substitute for `sh` even when `/bin/sh` is unusable. It's statically linked, and includes many standard utilities as built-ins (type "help" at the prompt for a reference list).

8.1.2 "I forgot the root password!" (2)

Boot from any emergency boot/root disk set. If */dev/hda3* is the original root partition, the following will let one edit the password file just as easily as the above.

```
# mkdir fixit
# mount /dev/hda3 fixit
# cd fixit/etc
# vi shadow
# vi passwd
```

The advantage of this approach over the previous method is one does not need to know the `lilo` password (if any). But to use it one must be able to access the BIOS setup to allow the system to boot from floppy disk or CD, if that is not already set.

8.1.3 Cannot boot the system

No problem, even if you didn't bother to make a boot disk during install. If `lilo` is broken, grab the boot disk from the Debian installation set and boot your system from it. At the boot prompt, assuming the root partition of your Linux installation is on */dev/hda12* and you want runlevel 3, enter:

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

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

If you need a custom boot floppy, follow `readme.txt` on the rescue disk.

8.1.4 Other boot tricks with the boot prompt

The system can be booted into a particular runlevel and configuration using the `lilo` boot prompt. Details are given in the BootPrompt-HOWTO (`http://www.tldp.org/HOWTO/BootPrompt-HOWTO.html`) (LDP).

If you want to boot the system into runlevel 4, use the following input at the `lilo` boot prompt.

```
boot: Linux 4
```

If you want to boot the system into normally functioning single-user mode and you know the root password, one of the following examples at the `lilo` boot prompt will work.

```
boot: Linux S
boot: Linux 1
boot: Linux -s
```

If you want to boot the system with less memory than system actually has (say 48MB for a system with 64MB), use this input at the `lilo` boot prompt:

```
boot: Linux mem=48M
```

Make sure not to specify more than the actual memory size here, otherwise the kernel will crash. If one has more than 64MB of memory, e.g. 128MB, unless one executes `mem=128M` at the boot prompt or includes a similar append line in `/etc/lilo.conf`, old kernels and/or a motherboard with an old BIOS will not use memory beyond 64MB.

8.2 Recording activities

8.2.1 Recording shell activities

System administration involves much more elaborate tasks in a Unix environment than in an ordinary personal computer environment. Make sure to know the most basic means of configuration in case you need to recover from system trouble. X-window-based GUI configuration tools look nice and convenient but are often unsuitable in these emergency situations.

In this context, recording shell activities is a good practice, especially as root.

Emacs: Use `M-x shell` to start recording into a buffer, and use `C-x C-w` to write the buffer to a file.

Shell: Use the `script` command.

```
$ script
Script started, file is typescript
 ... do whatever ...
 Control-D
$ col -bx <typescript >savefile
$ vi savefile
```

The following can be used instead of `script`:

```
$ bash -i 2>&1 | tee typescript
```

8.2.2 Recording X activities

If you need to record the graphic image of an X application, including an xterm display, use `gimp` (GUI). It can capture each window or the whole screen. Alternatives are `xwd` (`xbase-clients`), `import` (`imagemagick`), or `scrot` (`scrot`).

8.2.3 Recording changes to the configuration files

`Changetrack` will record changes to the configuration files in RCS archives. See `changetrack(1)`.

```
# apt-get install changetrack
# vi changetrack.conf
```

8.3 Copy and archive a whole subdirectory

8.3.1 Basic commands for copying a whole subdirectory

If you need to rearrange file structure, move content including file links by:

```
Standard method:
# cp -a /source/directory /dest/directory # requires GNU cp
# (cd /source/directory && tar cf - . ) | \
        (cd /dest/directory && tar xvfp - )
If a hard link is involved, a pedantic method is needed:
# cd /path/to/old/directory
# find . -depth -print0 | afio -p -xv -0a /mount/point/of/new/directory
If remote:
# (cd /source/directory && tar cf - . ) | \
        ssh user@host.dom (cd /dest/directory && tar xvfp - )
If there are no linked files:
# scp -pr user1@host1.dom:/source/directory \
          user2@host2.dom:/dest/directory
```

Here, `scp` <==> `rcp` and `ssh` <==> `rsh`.

The following comparative information on copying a whole subdirectory was presented by Manoj Srivastava <srivasta@debian.org> to debian-user@lists.debian.org.

8.3.2 **`cp`**

Traditionally, `cp` was not really a candidate for this task since it did not dereference symbolic links, or preserve hard links either. Another thing to consider was sparse files (files with holes).

GNU `cp` has overcome these limitations; however, on a non-GNU system, `cp` could still have problems. Also, you can't generate small, portable archives using `cp`.

```
% cp -a . newdir
```

8.3.3 **`tar`**

Tar overcame some of the problems that `cp` had with symbolic links. However, although `cpio` handles special files, traditional `tar` doesn't.

`tar`'s way of handling multiple hard links to a file places only one copy of the link on the tape, but the name attached to that copy is the *only* one you can use to retrieve the file; `cpio`'s way puts one copy for every link, but you can retrieve it using any of the names.

The `tar` command changed its option for `.bz2` files between Potato and Woody, so use `--bzip2` in scripts instead of its short form `-I` (Potato) or `-j` (Woody).

8.3.4 `pax`

The new, POSIX (IEEE Std 1003.2-1992, pages 380–388 (section 4.48) and pages 936–940 (section E.4.48)), all-singing, all-dancing, Portable Archive Interchange utility. `pax` will read, write, and list the members of an archive file, and will copy directory hierarchies. `pax` operation is independent of the specific archive format, and supports a wide variety of different archive formats.

`pax` implementations are still new and wet behind the ears.

```
# apt-get install pax
$ pax -rw -p e . newdir
 or
$ find . -depth  | pax -rw -p e  newdir
```

8.3.5 `cpio`

copies files into or out of a `cpio` or `tar` archive. The archive can be another file on the disk, a magnetic tape, or a pipe.

```
$ find . -depth -print0 | cpio --null --sparse -pvd new-dir
```

8.3.6 `afio`

`afio` is a better way of dealing with `cpio`-format archives. It is generally faster than cpio, provides more diverse magnetic tape options and deals somewhat gracefully with input data corruption. It supports multi-volume archives during interactive operation. `afio` can make compressed archives that are much safer than compressed `tar` or `cpio` archives. `afio` is best used as an "archive engine" in a backup script.

```
$ find . -depth -print0 | afio -px -0a new-dir
```

All my backups onto tape use `afio`.

8.4 System freeze recovery

8.4.1 Kill a process

Run `top` to see what process is acting funny. Press 'P' to sort by cpu usage, 'M' to sort by memory, and 'k' to kill a process.

Use `kill` to kill (or send a signal to) a process by process ID, `killall` to do the same by process command name. Frequently used signals:

```
   1: HUP,  restart daemon
  15: TERM, normal kill
   9: KILL, kill hard
```

8.4.2 ALT-SysRq

Insurance against system malfunction is provided by the kernel compile option "Magic SysRq key". Pressing `ALT-SysRq` on an i386, followed by one of the keys `r 0 k e i s u b`, does the magic.

Un'r'aw restores the keyboard after things like X crashes. Changing the console loglevel to '0' reduces error messages. sa'k' (system attention key) kills all processes on the current virtual console. t'e'rminate kills all processes on the current terminal except init. k'i'll kills all processes except init.

'S'ync, 'u'mount, and re'b'oot are for getting out of really bad situations.

Debian default installation kernels are not compiled with this option at the time this document is written. Recompile the kernel to activate this function. Detailed information is in `/usr/share/doc/kernel-doc-version/Documentation/sysrq.txt.gz` or `/usr/src/`*`kernel-version`*`/Documentation/sysrq.txt.gz`.

8.5 Nifty little commands to remember

8.5.1 Pager

`less` is the pager (file content browser). Hit 'h' for help. It can do much more than `more`. `less` can be supercharged by executing `eval $(lesspipe)` or `eval $(lessfile)` in the shell start-up script. See more in `/usr/share/doc/lessf/LESSOPEN`. The `-R` option allows raw character output and enables ANSI color escape sequences. See `less(1)`.

`w3m` may be a useful alternative pager for some code systems (EUC).

8.5.2 Free memory

`free` and `top` give good information on memory resources. Do not worry about the size of "used" in the "Mem:" line, but read the one under it (38792 in the example below).

```
$ free -k # for 256MB machine
             total       used       free     shared    buffers cached
Mem:        257136     230456      26680      45736     116136 75528
-/+ buffers/cache:      38792     218344
Swap:       264996          0     264996
```

The exact amount of physical memory can be confirmed by `grep '^Memory' /var/log/dmesg`, which in this case gives "Memory: 256984k/262144k available (1652k kernel code, 412k reserved, 2944k data, 152k init)".

```
Total         = 262144k = 256M (1k=1024, 1M=1024k)
Free to dmesg = 256984k = Total - kernel - reserved - data - init
Free to shell = 257136k = Total - kernel - reserved - data
```

About 5MB is not usable by the system because the kernel uses it.

8.5.3 Set time (BIOS)

```
# date MMDDhhmmCCYY
# hwclock --utc
# hwclock --systohc
# hwclock --show
```

This will set system and hardware time to MM/DD hh:mm, CCYY. Times are displayed in local time but hardware time uses UTC.

8.5.4 Set time (NTP)

Reference: Managing Accurate Date and Time HOWTO (`http://www.tldp.org/HOWTO/TimePrecision-H` `index.html`).

Set time with permanent Internet connection

Set system clock to the correct time automatically via a remote server:

```
# ntpdate server
```

This is good to have in `/etc/cron.daily` if your system has a permanent Internet connection.

Set time with sporadic Internet connection

Use the `chrony` package.

8.5.5 How to disable the screensaver

In the Linux console:

```
# setterm -powersave off
```

Start the kon2(kanji) console with:

```
# kon -SaveTime 0
```

While running X:

```
# xset s off
 or
# xset -dpms
 or
# xscreensaver-command -prefs
```

Read the corresponding manpages.

8.5.6 Search administrative database

Glibc offers `getent(1)` for searching entries from administrative databases, i.e., passwd, group, hosts, services, protocols, or networks.

```
getent database [key ...]
```

8.5.7 Disable sound (beep)

One can always unplug the PC speaker ;-) For the Bash shell:

```
echo "set bell-style none">> ~/.inputrc
```

8.5.8 Error messages on the console screen

In order to quiet on-screen error messages, the first place to check is `/etc/init.d/klogd`. Set `KLOGD="-c 3"` in this script and run `/etc/init.d/klogd restart`. An alternative method is to run `dmesg -n3`.

Here error levels mean:

- 0: KERN_EMERG, system is unusable
- 1: KERN_ALERT, action must be taken immediately
- 2: KERN_CRIT, critical conditions
- 3: KERN_ERR, error conditions
- 4: KERN_WARNING, warning conditions
- 5: KERN_NOTICE, normal but significant condition
- 6: KERN_INFO, informational
- 7: KERN_DEBUG, debug-level messages

If one particular useless error message bothers you a lot, consider making a trivial kernel patch like `shutup-abit-bp6` (available in the examples subdirectory (`examples/`)).

Another place to look may be `/etc/syslog.conf`; check to see whether any messages are logged to a console device.

8.5.9 Set console to the correct type

Console screens in Unix-like systems are usually accessed using (n)curses library routines. These give the user a terminal-independent method of updating character screens with reasonable optimization. See `ncurses(3X)` and `terminfo(5)`.

On a Debian system, there are quite a lot of predefined entries:

```
$ toe | less                    # all entries
$ toe /etc/terminfo/ | less     # user reconfigurable entries
```

Export your selection as environment variable `TERM`.

If the terminfo entry for xterm doesn't work with a non-Debian xterm, change your terminal type from "xterm" to one of the feature-limited versions such as "xterm-r6" when you log in to a Debian system remotely. See `/usr/share/doc/libncurses5/FAQ` for more. "dumb" is the lowest common denominator for terminfo.

8.5.10 Get the console back to a sane state

When the screen goes berserk after `$ cat some-binary-file` (you may not be able to see the command echoed as you type):

```
$ reset
```

8.5.11 Convert a text file from DOS to Unix style

Convert a DOS text file (end-of-line = `^M^J`) to a Unix text file (end-of-line = `^J`).

```
# apt-get install sysutils
$ dos2unix dosfile
```

8.5.12 Regular-expression substitution

Replace all instances of *FROM_REGEX* with *TO_REGEX* in all of the files *FILES* . . . :

```
# perl -i -p -e 's/FROM_REGEX/TO_REGEX/g;' FILES ...
```

`-i` is for "in-place editing", `-p` is for "implicit loop over *FILES* . . . ". If the substitution is complex, you can make recovery from errors easier by using the parameter `-i.bak` instead of `-i`; this will keep each original file, adding `.bak` as a file extension.

8.5.13 Convert a large file into small files

```
$ split -b 650m file   # split file into 650 MB chunks
$ cat x* >largefile    # merge files into 1 large file
```

8.5.14 Script snippets for piping commands

The following scripts will do nice things as a part of a pipe.

```
find /usr | egrep -v "/usr/var|/usr/tmp|/usr/local"
                      # find all files in /usr excluding some files
xargs -n 1 command    # run command for all items from stdin
xargs -n 1 echo|      # split white-space-separated items into lines
grep -e pattern|      # extract lines containing pattern
cut -d: -f3 -|        # extract third field separated by :
                      # (passwd file etc.)
col -bx |             # remove backspace and expand tabs to spaces
expand -|             # expand tabs
sort -u|              # sort and remove duplicates
```

```
tr '\n' ' '|          # concatenate lines into one line
tr '\r' ''|           # remove CR
tr 'A-Z' 'a-z'|       # convert uppercase to lowercase
sed 's/^/# /'|        # make each line a comment
sed 's/\.ext//g'|     # remove .ext
sed  -n -e 2p|        # print the second line
head -n 2 -|          # print the first 2 lines
tail -n 2 -|          # print the last 2 lines
```

8.5.15 Get text or a mailing list archive from a Web page

The following will read a Web page into a text file. Very useful when copying configurations off the Web.

```
$ lynx -dump http://www.remote-site.com/help-info.html >textfile
```

`links` and `w3m` can be used here, too, with slight differences in rendering.

If this is a mailing list archive, use `munpack` to obtain mime contents from text.

8.5.16 Pretty print a Web page

The following will read a Web page into a PostScript file/printer.

```
$ apt-get install html2ps
$ html2ps URL | lpr
```

See '`lpr/lpd`' on page 38. Also check `a2ps` and `mpage` packages for creating PostScript files.

8.5.17 Time a command

Display time used by a process.

```
# time some-command >/dev/null
real    0m0.035s       # time on wall clock (elapsed real time)
user    0m0.000s       # time in user mode
sys     0m0.020s       # time in kernel mode
```

8.5.18 `nice` command

Use `nice` (from the GNU `shellutils` package) to set a command’s nice value when starting. `renice` (`bsdutils`) or `top` can renice a process. A nice value of 19 represents the slowest (lowest priority) process; negative values are “not-nice”, with -20 being a very fast (high priority) process. Only the superuser can set negative nice values.

```
# nice  -19 top                                         # very nice
# nice --20 cdrecord -v -eject speed=2 dev=0,0 disk.img # very fast
```

Sometimes an extreme nice value does more harm than good to the system. Use this command carefully.

8.5.19 Schedule activity (`cron`, `at`)

Use `cron` and `at` to schedule tasks under Linux. See `at(1)`,`crontab(5)`,`crontab(8)`.

Run the command `crontab -e` to create or edit a crontab file to set up regularly scheduled events. Example of a crontab file:

```
# use /bin/sh to run commands, no matter what /etc/passwd says
SHELL=/bin/sh
# mail any output to ‘paul’, no matter whose crontab this is
MAILTO=paul
# Min Hour DayOfMonth Month DayOfWeek command (Day... are OR’ed)
# run at 00:05, every day
5  0  *  * *   $HOME/bin/daily.job >> $HOME/tmp/out 2>&1
# run at 14:15 on the first of every month -- output mailed to paul
15 14 1  * *   $HOME/bin/monthly
# run at 22:00 on weekdays(1-5), annoy Joe. % for newline, last % for cc:
0 22 *   * 1-5 mail -s "It’s 10pm" joe%Joe,%%Where are your kids?%.%%
23 */2 1 2 *   echo "run 23 minutes after 0am, 2am, 4am ..., on Feb 1"
5  4 *   * sun echo "run at 04:05 every sunday"
# run at 03:40 on the first Monday of each month
40 3 1-7 * *   [ "$(date +%a)" == "Mon" ] && command -args
```

Run the `at` command to schedule a one-time job:

```
$ echo ’command -args’| at 3:40 monday
```

8.5.20 Console switching with `screen`

The `screen` program allows you to run **multiple** virtual terminals, each with its own interactive shell, on a **single** physical terminal or terminal emulation window. Even if you use Linux virtual consoles or multiple xterm windows, it is worth exploring `screen` for its rich **feature set**, which includes

- scrollback history,
- copy-and-paste,
- output logging,
- digraph entry, and
- the ability to **detach** an entire `screen` session from your terminal and reattach it later.

Remote access scenario

If you frequently log on to a Linux machine from a remote terminal or using a VT100 terminal program, `screen` will make your life much easier with the **detach** feature.

1. You are logged in via a dialup connection, and are running a complex `screen` session with editors and other programs open in several windows.
2. Suddenly you need to leave your terminal, but you don't want to lose your work by hanging up.
3. Simply type `^A d` to **detach** the session, then log out. (Or, even quicker, type `^A DD` to have `screen` detach and log you out itself.)
4. When you log on again later, enter the command `screen -r`, and `screen` will magically **reattach** all the windows you had open.

Typical `screen` commands

Once you start `screen`, all keyboard input is sent to your current window except for the command keystroke, by default `^A`. All `screen` commands are entered by typing `^A` plus a single key [plus any parameters]. Useful commands:

```
^A ?     show a help screen (display key bindings)
^A c     create a new window and switch to it
^A n     go to next window
^A p     go to previous window
^A 0     go to window number 0
^A w     show a list of windows
^A a     send a Ctrl-A to current window as keyboard input
^A h     write a hardcopy of current window to file
^A H     begin/end logging current window to file
```

```
^A ^X    lock the terminal (password protected)
^A d     detach screen session from the terminal
^A DD    detach screen session and log out
```

This is only a small subset of `screen`'s commands and features. If there's something you want `screen` to be able to do, chances are it can! See `screen(1)` for details.

Backspace and/or Ctrl-H in `screen` session

If you find that backspace and/or Ctrl-H do not work properly when you are running `screen`, edit `/etc/screenrc`, find the line reading

```
bindkey -k kb stuff "\177"
```

and comment it out (i.e., add "#" as the first character).

Equivalent program to `screen` for X

Check out `xmove`. See `xmove(1)`.

8.5.21 Network testing basics

Install `netkit-ping`, `traceroute`, `dnsutils`, `ipchains` (for 2.2 Kernel), `iptables` (for 2.4 Kernel), and `net-tools` packages and:

```
$ ping yahoo.com            # check Internet connection
$ traceroute yahoo.com      # trace IP packets
$ ifconfig                  # check host config
$ route -n                  # check routing config
$ dig [@dns-server.com] host.dom [{a|mx|any}] |less
      # check host.dom DNS records  by dns-server.com
      # for a {mx|any} record
$ ipchains  -L -n |less     # check packet filter (2.2 kernel)
$ iptables -L -n |less      # check packet filter (2.4 kernel)
$ netstat -a                # find all open ports
$ netstat -l --inet         # find listening ports
$ netstat -ln --tcp         # find listening TCP ports (numeric)
```

8.5.22 Flush mail from local spool

To flush mail from the local spool:

```
# exim -q    # flush waiting mail
# exim -qf   # flush all mail
# exim -qff  # flush even frozen mail
```

`-qff` may be better as an option in the `/etc/ppp/ip-up.d/exim` script.

8.5.23 Remove frozen mail from local spool

To remove frozen mail from the local spool with a delivery error message:

```
# exim -Mg `mailq | grep frozen | awk '{ print $3 }'`
```

8.5.24 Clear file contents

In order to clear the contents of a file such as a logfile, do not use `rm` to delete the file and then create a new empty file, because the file may still be accessed in the interval between commands. The following is the safe way to clear the contents of the file.

```
$ :>file-to-be-cleared
```

8.5.25 Dummy files

The following commands will create dummy or empty files:

```
$ dd if=/dev/zero    of=filename bs=1k count=5 # 5KB of zero content
$ dd if=/dev/urandom of=filename bs=1m count=7 # 7MB of random content
$ touch filename #  create 0B file (if file exists, updates mtime)
```

8.5.26 `chroot`

Suppose you have installed a whole Linux distribution (this can be another release of Debian or even a Red Hat distribution) into *`/dev/hda1`* and you are running another Linux system installed in *`/dev/hda2`*. You can execute the system in *`/dev/hda1`* without rebooting the running system in *`/dev/hda2`* by sharing the same kernel.

```
# mount /dev/hda1 /mnt/target
 ... supposing /dev/hda1 contains one whole system
# chroot /mnt/target
 ... Now the contents of /dev/hda1 are seen as the root directory.
# mount proc /proc              # just in case
 ... run commands within /dev/hda1
```

This enables stable/testing/unstable to be loaded on one machine. Also, one may run a memory-hungry program such as `dselect` by running it on a host machine while NFS-mounting a satellite machine to the host as r/w and pointing `chroot` to the satellite machine.

A `chroot` system can easily be created by the `debootstrap` command in Woody.

```
# mkdir potatochroot
# debootstrap potato potatochroot
# chroot potatochroot
# apt-setup # set-up /etc/apt/sources.list
```

There is a more specialized chroot package, `pbuilder`, which constructs a chroot system and builds a package inside the chroot. It is an ideal system to use to check that a package's build-dependencies are correct, and to be sure that unnecessary and wrong build dependencies will not exist in the resulting package.

8.5.27 `mount` hard disk image file

If *`file.img`* contains an image of hard disk contents and the original hard disk had a disk configuration which gives *xxxx* = (bytes/sector) * (sectors/cylinder), then the following will mount it to `/mnt`:

```
# mount -o loop,offset=xxxx file.img /mnt
```

Note that most hard disks have 512 bytes/sector.

8.5.28 Samba

Basics of getting files from Windoze:

```
# mount -t smbfs -o username=myname,uid=my_uid,gid=my_gid \
        //server/share /mnt/smb  # mount Windows files to Linux
# smbmount //server/share /mnt/smb \
        -o "username=myname,uid=my_uid,gid=my_gid"
# smbclient -L 192.168.1.2 # list the shares on a computer
```

Samba neighbors can be checked from Linux using:

```
# smbclient -N -L ip_address_of_your_PC | less
# nmblookup -T "*"
```

Chapter 9

Tuning a Debian system

This chapter describes only the basics of system configuration. A prerequisite of this chapter is reading 'Debian System installation hints' on page 27.

For the security concious, it is highly recommended to read the Securing Debian Manual (`http://www.debian.org/doc/manuals/securing-debian-howto/`), which can also be found as the `harden-doc` package.

9.1 System initialization hints

See 'The `init` program' on page 22 for the basics of the Debian init script.

9.1.1 Customizing init scripts

Debian uses the sys-V init script system. Although all init scripts in `/etc/init.d/*` are marked as conffiles and sysadmins are free to modify them, customizing init scripts by editing files in `/etc/defaults/*` is the preferred approach.

For example, `/etc/init.d/rcS` can be used to customize boot-time defaults for `motd`, `sulogin`, etc.

9.1.2 Customizing system logging

System log mode can be configured using `/etc/syslog.conf`. Check the `colorize` package for a program to colorize system log files. See also `syslogd(8)` and `syslog.conf(5)`.

9.1.3 Hardware access optimization

There are a few hardware optimization configurations that Debian leaves to the sysadmin to take care of.

- `hdparm`
  - Hard disk access optimization. Very effective.
  - Dangerous. You must read `hdparm(8)` first.
  - `hdparm -tT` *`/dev/hda`* to test disk access speed.
  - `hdparm -c1 -d1 -u1 -m16 -A` *`/dev/hda`* to speed up a modern IDE system. (It may be dangerous.)

- `setserial`
  - Collection of tools for serial port management.

- `scsitools`
  - Collection of tools for SCSI hardware management.

- `memtest86`
  - Collection of tools for memory hardware management.

- `hwtools`
  - Collection of tools for low-level hardware management.
    - irqtune: changes the IRQ priority of devices to allow devices that require high priority and fast service (e.g. serial ports, modems) to have it.
    - scanport: scans I/O space from 0x100 to 0x3ff looking for installed ISA devices.
    - inb: a quick little hack that reads an I/O port and dumps the value in hex and binary.

- `schedutils`
  - Linux scheduler utilities.
  - `taskset`, `irqset`, `lsrt`, and `rt` are included.
  - Together with `nice` and `renice` (not included), they allow full control of process scheduling parameters.

9.2 Access control

9.2.1 Access control through PAM and login

PAM (Pluggable Authentication Modules) provides login control.

```
/etc/pam.d/*              # PAM control files
/etc/pam.d/login          # PAM control file for login
/etc/security/*           # PAM module parameters
/etc/securetty            # this controls root login by console (login)
/etc/login.defs           # this controls login behaviors (login)
```

Change the contents of `/etc/pam.d/login` as follows, if you want insecure but passwordless console terminals at your own risk.

```
#auth       required   pam_unix.so nullok
auth       required   pam_permit.so
```

Similar tricks can be applied for `xdm`, `gdm`, . . . , for passwordless console X.

The maximum number of processes can be set with `ulimit -u 1000` in a Bash shell or with settings in `/etc/security/limits.conf` from PAM. Other parameters such as `core` can be set similarly. The initial value of `PATH` can be set by `/etc/login.defs` before the shell start up script.

The documentation for PAM is packaged in the `libpam-doc` package. The Linux-PAM System Administrator's Guide covers configuring PAM, what modules are available etc. The documentation also includes The Linux-PAM Application Developers' Guide and The Linux-PAM Module Writers' Guide.

9.2.2 "Why GNU `su` does not support the `wheel` group"

This is the famous phrase at the bottom of the old `info su` page by Richard M. Stallman. Not to worry: the current `su` in Debian uses PAM, so that one can restrict the ability to use `su` to any group using `pam_wheel.so` in `/etc/pam.d/su`. The following will set the `adm` group in a Debian system as an equivalent of the BSD `wheel` group and allow `su` without a password for its members.

```
# anti-RMS configuration in /etc/pam.d/su
auth       required   pam_wheel.so group=adm

# Wheel members to be able to su without a password
auth       sufficient pam_wheel.so trust group=adm
```

9.2.3 Meaning of various groups

A few interesting groups:

- `root` group is the default wheel group for `su` if `pam_wheel.so` is used without the `group=` argument.
- `adm` group can read logfiles.
- `cdrom` group can be used locally to give a set of users access to a CD-ROM drive.
- `floppy` group can be used locally to give a set of users access to a floppy drive.
- `audio` group can be used locally to give a set of users access to an audio device.
- `src` group owns source code, including files in `/usr/src`. It can be used locally to give a user the ability to manage system source code.
- `staff` membership is useful for helpdesk types or junior sysadmins, giving them the ability to do things in `/usr/local` and to create directories in `/home`.

For a complete list, see the "FAQ" section in the Securing Debian Manual (`http://www.debian.org/doc/manuals/securing-debian-howto/`), which can also be found as the `harden-doc` package.

9.2.4 `sudo` – a safer work environment

My usage of `sudo` is mostly a protection from my own stupidity. I consider using `sudo` a better alternative to always using the system as root. YMMV.

Install `sudo` and activate it by setting options in /etc/sudoers (`examples/`). Also check out the `sudo` group feature in `/usr/share/doc/sudo/OPTIONS`.

The sample configuration provides "staff" group members access to any commands run as root under `sudo` and also gives "src" members access to selected commands run as root under `sudo`.

The advantage of `sudo` is that it only requires an ordinary user's password to log in, and activity is monitored. This is a nice way to give some authority to a junior administrator. For example:

```
$ sudo chown -R myself:mygrp .
```

Of course if you know the root password (as most home users do), any command can be run under root from a user account:

```
$ su -c "shutdown -h now"
Password:
```

(I know I should tighten the admin account's `sudo` privileges. But since this is my home server, I have not bothered yet.)

For a different program that allows ordinary users to run commands with root privileges, see the `super` package.

9.2.5 Access control to daemon programs

The Internet *super-server*, `inetd`, is started at boot time by `/etc/rc2.d/S20inetd` (for RUNLEVEL=2), which is a symlink to `/etc/init.d/inetd`. Essentially, `inetd` allows running one daemon to invoke several others, reducing load on the system.

Whenever a request for service arrives, its protocol and service are identified by looking them up in the databases in `/etc/protocols` and `/etc/services`. `inetd` then looks up a normal Internet service in the `/etc/inetd.conf` database, or a Sun-RPC based service in `/etc/rpc.conf`.

For system security, make sure to disable unused services in `/etc/inetd.conf`. Sun-RPC services need to be active for NFS and other RPC-based programs.

Sometimes, `inetd` does not start the intended server directly but starts the `tcpd` TCP/IP daemon wrapper program with the intended server name as its argument in `/etc/inetd.conf`. In this case, `tcpd` runs the appropriate server program after logging the request and doing some additional checks using `/etc/hosts.deny` and `/etc/hosts.allow`.

If you have problem with remote access in a recent Debian system, comment out "ALL: PARANOID" in `/etc/hosts.deny` if it exists.

For details, see `inetd(8)`, `inetd.conf(5)`, `protocols(5)`, `services(5)`, `tcpd(8)`, `hosts_access(5)`, and `hosts_options(5)`.

For more information on Sun-RPC, see `rpcinfo(8)` , `portmap(8)`, and `/usr/share/doc/portmap/portmapper.txt.gz`.

9.2.6 Lightweight Directory Access Protocol

References:

- OpenLDAP (`http://www.openldap.org/`)
- OpenLDAP Admin Guide in the `openldap-guide` package
- LDP: LDAP Linux HOWTO (`http://www.tldp.org/HOWTO/LDAP-HOWTO/index.html`)
- LDP: LDAP Implementation HOWTO (`http://www.tldp.org/HOWTO/LDAP-Implementation-HOWT` `index.html`)
- OpenLDAP, extensive use reports (`http://portal.aphroland.org/~aphro/ldap-docs/ldap.html`)
- Open LDAP with Courier IMAP and Postfix (`http://annapolislinux.org/docs/plc/postfix-courier-howto.txt`)

9.3 CD-writer

CD-writers with ATAPI/IDE interfaces have recently become a very popular option. It is a nice medium for system backup and archiving for the home user needing < 640 MB capacity. For the most authoritative information, see the LDP CD-Writing-HOWTO (`http://www.tldp.org/HOWTO/CD-Writing-HOWTO.html`).

9.3.1 Introduction

First, any disruption of data sent to the CD-writer will cause irrecoverable damage to the CD. Get a CD-writer with as large a buffer as possible. If money is no object, do not bother with ATAPI/IDE, just get a SCSI version. If you have a choice of IDE interface to be connected, use the one on the PCI-bus (i.e., on the motherboard) rather than one on the ISA-bus (an SB16 card, etc.).

When a CD-writer is connected to IDE, it has to be driven by the IDE-SCSI driver instead of an ordinary IDE CD driver. Also, the SCSI generic driver needs to be activated. There are two possible approaches to doing this, assuming a kernel distributed with modern distributions (as of March 2001).

9.3.2 Approach 1: modules + `lilo`

Add the following line to `/etc/lilo.conf` if you are using a stock Debian kernel. If multiple options are used, list them separated by spaces:

```
append="hdx=ide-scsi ignore=hdx"
```

Here the location of the CD-writer, which is accessed through the ide-scsi driver, is indicated by `hdx`, where *x* represents one of the following:

```
hdb          for a slave on the first IDE port
hdc          for a master on the second IDE port
hdd          for a slave on the second IDE port
hde ... hdh  for a drive on an external IDE port
```

Type the following commands as root to activate after finishing all the configuration:

```
# lilo
# shutdown -h now
```

9.3.3 Approach 2: recompile the kernel

Debian uses `make-kpkg` to create a kernel. Use the new `--append_to_version` with make-kpkg to build multiple kernel images. See 'The Linux kernel under Debian' on page 73.

Use the following setup through `make menuconfig`:

- bzImage
- Exclude the IDE CD driver (not a must, but simpler to do this)
- Compile in ide-scsi and sg, or make them modules

9.3.4 Post-configuration steps

Kernel support for the CD-writer can be activated during booting by the following:

```
# echo ide-scsi >>/etc/modules
# echo sg       >>/etc/modules
# cd /dev; ln -sf scd0 cdrom
```

Manual activation can be done by:

```
# modprobe ide-scsi
# modprobe sg
```

After reboot, you can check installation by:

```
$ dmesg|less
# apt-get install cdrecord
# cdrecord -scanbus
```

[Per Warren Dodge] Sometimes there may be conflicts between `ide-scsi` and `ide-cd` if there are both CD-ROM and CD-R/RW on the system. Try adding the following line to your `/etc/modutils/aliases`, running `update-modules`, and rebooting.

```
pre-install      ide-scsi      modprobe ide-cd
```

This causes the IDE driver to load before `ide-scsi`. The IDE driver `ide-cd` takes control of the ATAPI CD-ROM — anything that it hasn't been told to **ignore**. That leaves just the ignored devices for `ide-scsi` to control.

9.3.5 CD-image file (bootable)

To create a CD-ROM of files under `target-directory/` as `cd-image.raw` (bootable, Joliet TRANS.TBL-enabled format; if not bootable, take out `-b` and `-c` options), insert a boot floppy in the first floppy drive and

```
# dd if=/dev/fd0 target-directory/boot.img
# mkisofs -r -V volume_id -b boot.img -c bootcatalog -J -T \
        -o cd-image.raw target_directory/
```

One funny hack is to make a bootable DOS CD-ROM. If an ordinary DOS boot floppy disk image is in the above *`boot.img`*, the CD-ROM will boot as if a DOS floppy were in the first floppy drive (A:). Doing this with freeDOS may be more interesting.

This CD-image file can be inspected by mounting it on the loop device.

```
# mount -t iso9660 -o ro,loop cd-image.raw /cdrom
# cd /cdrom
# mc
# umount /cdrom
```

9.3.6 Write to the CD-writer (R, R/W):

First test with (assuming double speed)

```
# nice --10 cdrecord -dummy speed=2 dev=0,0 disk.img
```

Then if OK, write to CD-R with

```
# nice --10 cdrecord -v -eject speed=2 dev=0,0 disk.img
```

Or write to a CD-RW disk with

```
# nice --10 cdrecord -v -eject blank=fast speed=2 dev=0,0 disk.img
```

Some CD-RW drives work better with

```
# nice --10 cdrecord -v blank=all speed=2 dev=0,0
```

followed by

```
# nice --10 cdrecord -v -eject speed=2 dev=0,0 disk.img
```

Two steps are needed to prevent SCSI timeouts during blanking from interfering with the burning step. The argument value to `nice` may require some adjustments.

9.3.7 Make an image file of a CD

Some CD-Rs and commercial CDs have junk sectors at the end that make copying by `dd` impossible (the Windows 98 CD is one of them). The `cdrecord` package comes with the `readcd` command. Use this to copy any CD contents to an image file. If it is a data disk, mount it and run `mount` to see its actual size. Divide the number shown (in blocks, = 1024 bytes) by 2 to get the number of actual CD sectors (2048 bytes). Run `readcd` with options and use this disk image to burn the CD-R/RW.

```
# readcd target lun scsibusno # select function 11
```

Here, set all 3 command-line parameters to 0 for most cases. Sometimes the number of sectors given by `readcd` is excessive! Use the above number from an actual mount for better results.

```
My CD-R        = +2 sectors
MS Windows CD = +1 sector, i.e., +2048 bytes
```

9.3.8 Debian CD images

To obtain the latest information on Debian CDs, visit the Debian CD site (`http://www.debian.org/CD/`).

If you have a fast Internet connection, think about installing over the network using:
- a few floppy images (`http://www.debian.org/distrib/floppyinst`).
- a minimal bootable CD image (`http://www.debian.org/CD/netinst/`).

If you do not have a fast Internet connection, think about purchasing CDs from a CD vendors (`http://www.debian.org/CD/vendors/`).

Please do not waste bandwidth by downloading standard CD images unless you are a CD image tester (even with the new jigdo method).

One noteworthy CD image is KNOPPIX - Live Linux Filesystem On CD (`http://www.knopper.net/knoppix/index-en.html`). This CD will boot a functioning Debian system without installing itself to the hard disk.

9.3.9 Back up the system to CD-R

To copy key configuration files and data files to CD-R, use the example "backup" script backup (`examples/`).

9.3.10 Copy a music CD to CD-R

Not tested by me:

```
# apt-get install cdrecord cdparanoia
# cdparanoia -s -B
# cdrecord dev=0,0,0 speed=2 -v -dao -eject defpregap=1 -audio *.wav
```

or,

```
# apt-get install cdrdao #disk at once
# cdrdao read-cd --device /dev/cdrom --paranoia-mode 3 my_cd # read cd
# cdrdao write --device /dev/cdrom --speed 8 my_cd    # write a new CD
```

`cdrdao` does a real copy (no gaps, etc. . .)

9.4 The X program

'X server' on the facing page a program on a local host that displays an X window and/or desktop on a user's monitor (CRT, LCD) and accepts keyboard and mouse input.

'X client' on page 108 a program on a (local or remote) host that runs X-window-compatible application software.

This reverses the ordinary use of "server" and "client" in other contexts. For basics refer to `X(7)`, the LDP XWindow-User-HOWTO (`http://www.tldp.org/HOWTO/XWindow-User-HOWTO.html`), and the Remote X Apps mini-HOWTO (`http://www.tldp.org/HOWTO/mini/Remote-X-Apps.html`).

There are several ways of getting the "X server" (display side) to accept remote connections from an "X client" (application side):

- `xhost`
  - the host list mechanism (insecure).

  - non-encrypted protocol (prone to eavesdropping attack).
  - Do not use this, if possible.
  - See 'Remote X connection: `xhost`' on page 109 and `xhost(1x)`.

- `xauth`
  - the MIT magic cookie mechanism (insecure but better than `xhost`).
  - non-encrypted protocol (prone to eavesdropping attack).
  - use this only for local connection.
  - See 'Gain root in X' on page 110 and `xauth(1x)`.

- `xdm`, `wdm`, `gdm`, `kdm`, ...
  - See `xdm(1x)` and `Xsecurity(7)` for the basics of X display access control.
  - See `wdm(1x)`, `gdm(8)`, and `kdm.options(5)` for more information, if these are installed.
  - See 'System-V `init` and runlevels' on page 71 for how to disable `xdm` to gain a Linux console upon boot without purging the `xdm` package.

- `ssh -X`
  - port forwarding mechanism through secure shell (secure).
  - encrypted protocol (a waste of resources if used locally).
  - use this for remote connections.
  - See 'Remote X connection: `ssh`' on page 109.

All remote connection methods, except `ssh`, require TCP/IP connection enabled on the X server. See 'TCP/IP connection to X' on the following page.

9.4.1 X server

See `XFree86(1x)` for X server information.

To (re)configure X4 in Woody, run:

```
# dpkg-reconfigure --p=low xserver-xfree86
```

Invoke X server from a local console:

```
$ startx -- :<display> vtXX
e.g.:
$ startx -- :1 vt8
... start on vt8 connected to localhost:1
```

9.4.2 X client

Most X client programs can be started with a command like this:

```
client $ xterm -geometry 80x24+30+200 -fn 6x10 -display hostname:0 &
```

Here, the optional command-line arguments mean:

- `-geometry` *`WIDTH`*x*`HEIGHT`*+*`XOFF`*+*`YOFF`*: the initial size and location of the window.
- `-fn` *`FONTNAME`*: the font to use for displaying text. *`FONTNAME`* can be:
  - a14: Normal size font
  - a24: Large size font
  - … (check available fonts with `xlsfont`.)
- `-display` *`displayname`*: the name of the X server to use. *`displayname`* can be:
  - *`hostname`*`:`*`D.S`* means screen *`S`* on display *`D`* of host *`hostname`*; the X server for this display is listening to TCP port 6000+D.
  - *`host`*`/unix:`*`D.S`* means screen *`S`* on display *D* of host *`host`*; the X server for this display is listening to UNIX domain socket `/tmp/.X11-unix/XD` (so it's only reachable from *`host`*).
  - *`:D.S`* is equivalent to *`host`*`/unix:`*`D.S`*, where *`host`* is the local hostname.

The default *`displayname`* for the X client program (application side) can be set by the DISPLAY environment variable. For example, prior to running an X client program, executing one of the following commands achieves this:

```
$ export DISPLAY=:0
        # The default, local machine using the first X screen
$ export DISPLAY=hostname.fulldomain.name:0.2
$ export DISPLAY=localhost:0
```

9.4.3 TCP/IP connection to X

Because a remote TCP/IP socket connection without encryption is prone to an eavesdropping attack, the default setting for X in recent Debian versions disables the TCP/IP socket. Consider using `ssh` for a remote X connection (see 'Remote X connection: `ssh`' on the next page).

The method described here is not encouraged unless one is in a very secure environment behind a good firewall system with only trusted users present. Use the following command to verify your current X server setting for the TCP/IP socket:

```
# find /etc/X11 -type f -print0 | xargs -0 grep nolisten
/etc/X11/xinit/xserverrc:exec /usr/bin/X11/X -dpi 100 -nolisten tcp
```

Remove `-nolisten` to restore TCP/IP listening on the X server.

9.4.4 Remote X connection: **`xhost`**

`xhost` allows access based on hostnames. This is very insecure. The following will disable host checking and allow connections from anywhere if a TCP/IP socket connection is allowed (see 'TCP/IP connection to X' on the facing page):

```
$ xhost +
```

You can re-enable host checking with:

```
$ xhost -
```

`xhost` does not distinguish between different users on the remote host. Also, hostnames (addresses actually) can be spoofed.

This method must be avoided even with more restrictive host criteria if you're on an untrusted network (for instance with dialup PPP access to the Internet). See `xhost(1x)`.

9.4.5 Remote X connection: **`ssh`**

The use of `ssh` enables a secure connection from a local X server to a remote application server.

- Set `X11Forwarding` and `AllowTcpForwarding` entries to `yes` in `/etc/ssh/sshd_config` of the remote host.
- Start the X server on the local host.
- Open an `xterm` in the local host.
- Run `ssh` to establish a connection with the remote site.

```
localname @ localhost $ ssh -q -X -l loginname remotehost.domain
Password:
.....
```

- Run X application commands on the remote site.

```
loginname @ remotehost $ gimp &
```

This method allows the display of the remote X client output as if it were locally connected through a local UNIX domain socket.

9.4.6 `xterm`

Learn everything about `xterm` at `http://dickey.his.com/xterm/xterm.faq.html`.

9.4.7 Gain root in X

If a GUI program needs to be run with root privilege, use the following procedures to display program output on a user's X server. Never attempt to start an X server directly from the root account in order to prevent possible security risks.

Start the X server as a normal user and open an `xterm` console. Then:

```
$ XAUTHORITY=$HOME/.Xauthority
$ export XAUTHORITY
$ su root
Password:*****
# printtool &
```

When using this trick to `su` to a non-root user, make sure `$HOME/.Xauthority` is group readable by this non-root user.

This command sequence can be automated by adding a few files. From the root account, create the file `/etc/X11/Xsession.d/00xfree86-common_environment`, containing the following lines:

```
if [ -f "$HOME/.xenvironment" ]; then
    . $HOME/.xenvironment
fi
```

From the user account, create the file `$HOME/.xenvironment` in the user's home directory, containing following lines:

```
# This makes X work when I su to the root account.
if [ -z "$XAUTHORITY" ]; then
        XAUTHORITY=$HOME/.Xauthority
        export XAUTHORITY
fi
```

Then run `su` (not `su -`) in an `xterm` window of the user. Now GUI programs started from this `xterm` can display output on this user's X window while running with root privilege. This trick works as long as the default Xsession is executed. If a user set up his customization using `$HOME/.xinit` or `$HOME/.xsession`, the abovementioned environment `XAUTHORITY` variable needs to be set similaly in those scripts.

Alternatively, `sudo` can be used to automate the command sequence:

```
$ sudo xterm
... or
$ sudo -H -s
```

Here `/root/.bashrc` should contain:

```
if [ $SUDO_USER ]; then
    sudo -H -u $SUDO_USER xauth extract - $DISPLAY | xauth merge -
fi
```

This works fine even with the home directory of the user on an NFS mount, because root does not read the `.Xauthority` file.

There are also several specialized packages for this purpose: `kdesu`, `gksu`, `gksudo`, `gnome-sudo`, and `xsu`. Some other methods can be used to achieve similar results: creating a symlink from `/root/.Xauthority` to the user's corresponding one; use of the script sux (`http://fgouget.free.fr/sux/sux-readme.shtml`); or putting "`xauth merge ~`*USER_RUNNING_X*`/.Xauthority`" in the root initialization script.

See more on the debian-devel mailing list (`http://lists.debian.org/debian-devel/2002/debian-devel-200207/msg00259.html`).

9.4.8 TrueType fonts in X

The standard `xfs` in XFree86-4 works fine with TrueType fonts. You have to install a third-party font server such as `xfs-xtt`, if you are using XFree86-3.

You just need to make sure that whatever apps you want to use the TrueType fonts are linked against libXft or libfreetype (you probably don't even have to worry about this if you're using precompiled .debs).

Remember to install required font files and generate the `fonts.{scale,dir}` files so that the fonts can be indexed and used.

Since **Free** fonts are sometimes limited, installing or sharing some commercial TrueType fonts is an option for a Debian users. In order to make this process easy for the user, some convienience packages have been created:

- `ttf-commercial`
- `msttcorefonts` (No longer useful as of 8/2002 due to M$ policy change)

You'll have a really good selection of TT fonts at the expense of contaminating your **Free** system with non-Free fonts.

9.4.9 Web Browser (graphical)

There are a few Web browser packages with graphical display capabilities as of the Woody release:

- `mozilla` The Mozilla browser (new)
- `galeon` Mozilla-based browser with a Gnome UI (new)
- `konqueror` KDE browser
- `amaya-gtk` W3C reference browser
- `amaya-lesstif` W3C reference browser
- `netscape-...` (many, old)
- `communicator-...` (many, old)
- ...

The version of `mozilla` must match the version that `galeon` requires. Although they differ in UI, these two programs share the Gecko HTML rendering engine.

Plug-ins for browsers such as `mozilla` and `galeon` can be enabled by installing "`*.so`" manually in the plug-in directory and restarting the browsers.

Plug-in resources:

- Java plug-in: install binary "J2SE" from `http://java.sun.com`.
- Flash plug-in: install binary "Macromedia Flash Player 5" from `http://www.macromedia.com/software/flashplayer/`.
- `freewrl`: VRML browser and Netscape plugin
- ...

9.4.10 CJK and X

Reference:

- 'Localization and national language support' on page 119

- SuSE pages for CJK (`http://www.suse.de/~mfabian/suse-cjk/suse-cjk.html`)

Here, let us try for Japanese:

- install Japanese packages:
  - `kinput2-canna-wnn` — An input server for X11 applications.
  - `kterm`, `mlterm`, and `jfbterm`: Japanese-compatible terminals.
  - `egg` — Input Method Architecture for Emacsen.
  - `canna` — A Japanese input system (server and dictionary).
  - `freewnn-jserver` — Network-extensible Kana-to-Kanji conversion system.
  - . . . and all the Japanese font packages.

  In reality, use `tasksel` or `aptitude` to select "Japanese environment" and avoid installing software that conflicts with the normal system.

- add a locale that supports Japanese characters (e.g., ja_JP.UTF-8; see 'Localization and national language support' on page 119).

- add the following environment values in `~/.xenvironment` using the same trick used in 'Gain root in X' on page 110.

  ```
  XMODIFIERS="@im=kinput2"
  LC_CTYPE=ja_JP.UTF-8 # some Japanese locale
  ```

  (Or manually do so in `xterm` before starting an application.)

- activate XIM `kinput2` by adding `*inputMethod: kinput2` to your X resources file (looks like Debian takes care of this).

- Some applications (such as `mlterm`) also allow you to set up `*inputMethod:` information dynamically at runtime (press Ctrl-MouseButton-3 in mlterm).

Once you have started the application, press "**Shift+Space**" and a window should pop up stating that you now may input Japanese characters.

9.5 SSH

SSH (Secure SHell) is the secure way to connect over the Internet. A free version of SSH called OpenSSH is available as the `ssh` package in Debian.

9.5.1 Basics

First install the OpenSSH server and client.

```
# apt-get update && apt-get install ssh
```

The non-US entry in the `/etc/apt/source.list` is required. `/etc/ssh/sshd_not_to_be_run` must not be present if one wishes to run the OpenSSH server.

SSH has 2 authentication protocols:

- SSH protocol version 1:
  - Potato version only supports this protocol.
  - available authentication methods:
    - RSAAuthentication: RSA identity key based user authentication
    - RhostsAuthentication: .rhosts based host authentication (insecure, disabled)
    - RhostsRSAAuthentication: .rhosts authentication combined with RSA host key (disabled)
    - ChallengeResponseAuthentication: RSA challenge-response authentication
    - PasswordAuthentication: password based authentication
- SSH protocol version 2:
  - post-Woody versions use this as primary protocol.
  - available authentication methods:
    - PubkeyAuthentication: public key based user authentication
    - HostbasedAuthentication: `.rhosts` or `/etc/hosts.equiv` authentication combined with public key client host authentication (disabled)
    - ChallengeResponseAuthentication: challenge-response authentication
    - PasswordAuthentication: password based authentication

Be careful about these differences if you are migrating to Woody or using a non-Debian system.

See `/usr/share/doc/ssh/README.Debian.gz`, `ssh(1)`, `sshd(8)`, `ssh-agent(1)`, and `ssh-keygen(1)` for details.

Following are the key configuration files:

- `/etc/ssh/ssh_config`: SSH client defaults. See `ssh(1)`. Notable entries are:
  - `Host`: Restricts the following declarations (up to the next Host keyword) to be only for those hosts that match one of the patterns given after the keyword.
  - `Protocol`: Specifies the SSH protocol versions. The default is “2,1”.
  - `PreferredAuthentications`: Specifies the SSH2 client authentication method. The default is “hostbased,publickey,keyboard-interactive,password”.

  - `ForwardX11`: The default is disabled. This can be overridden by the command-line option "`-X`".

- `/etc/ssh/sshd_config`: SSH server defaults. See `sshd(8)`. Notable entries are:
  - `ListenAddress`: Specifies the local addresses `sshd` should listen on. Multiple options are permitted.
  - `AllowTcpForwarding`: The default is disabled.
  - `X11Forwarding`: The default is disabled.

- `$HOME/.ssh/authorized_keys`: the lists of the default public keys that clients used to connect to this account on this host. See `ssh-keygen(1)`.

- `$HOME/.ssh/identity`: See `ssh-add(1)` and `ssh-agent(1)`.

The following will start an `ssh` connection from a client.

```
$ ssh username@hostname.domain.ext
$ ssh -1 username@hostname.domain.ext # Force SSH version 1
```

For the user, `ssh` functions as a smarter and more secure `telnet` (will not bomb with ^]).

9.5.2 Port forwarding – for SMTP/POP3 tunneling

To establish a pipe to connect to port 25 of *remote-server* from port 4025 of localhost, and to port 110 of *remote-server* from port 4110 of localhost through `ssh`, execute on the local machine:

```
# ssh -q -L 4025:remote-server:25 4110:remote-server:110 \
           username@remote-server
```

This is a secure way to make connections to SMTP/POP3 servers over the Internet. Set the `AllowTcpForwarding` entry to `yes` in `/etc/ssh/sshd_config` of the remote host.

9.5.3 Connect with fewer passwords

One can avoid having to remember a password for each remote system by using RSAAuthentication (SSH1 protocol) or PubkeyAuthentication (SSH2 protocol).

On the remote system, set respective entries, "RSAAuthentication yes" or "PubkeyAuthentication yes", in `/etc/ssh/sshd_config`.

Then generate authentication keys locally and install the public key on the remote system:

```
$ ssh-keygen          # RSAAuthentication: RSA1 key for SSH1
$ cat .ssh/id_rsa.pub | ssh user1@remote \
        "cat - >>.ssh/authorized_keys"
...
$ ssh-keygen -t rsa   # PubkeyAuthentication: RSA key for SSH2
$ cat .ssh/id_rsa.pub | ssh user1@remote \
        "cat - >>.ssh/authorized_keys"
...
$ ssh-keygen -t dsa   # PubkeyAuthentication: DSA key for SSH2
$ cat .ssh/id_dsa.pub | ssh user1@remote \
        "cat - >>.ssh/authorized_keys"
```

One can change the passphrase later with "`ssh-keygen -p`". Make sure to verify settings by testing the connection. In case of any problem, use "`ssh -v`".

You can add options to the entries in `authorized_keys` to limit hosts and to run specific commands. See `sshd(8)` for details.

Note that SSH2 has `HostbasedAuthentication`. For this to work, you must adjust settings of `HostbasedAuthentication` to `yes` in both `/etc/ssh/sshd_config` on the server machine and `/etc/ssh/ssh_config` or `$HOME/.ssh/config` on the client machine.

9.5.4 Foreign SSH clients

There are a few free SSH clients available for non-Unix-like platforms.

Windows puTTY (`http://www.chiak.greenend.org.uk/~sgtatham/putty/`) (GPL)

Windows (cygwin) SSH in cygwin (`http://www.cygwin.com/`) (GPL)

Macintosh Classic macSSH (`http://www.macssh.com/`) (GPL) [Note that Mac OS X includes OpenSSH; use `ssh` in the Terminal application]

See also SourceForge.net, site documentation (`http://www.sourceforge.net/docman/?group_id=1`), "6. CVS Instructions".

9.5.5 SSH agent

Just put your public key into `~/.ssh/authorized_keys`, and you're all set:

```
$ ssh-agent
$ # paste the output to your shell
$ ssh-add .ssh/identity
$ # or ssh-add .ssh/id_dsa or whatever your private key is named
$ scp remote.host.with.public.key
```

For more, read `ssh-agent(1)` and `ssh-add(1)`.

9.5.6 Troubleshooting

If you have problems, check the permissions of configuration files and run `ssh` with the "`-v`" option.

Use the "`-P`" option if you are root and have trouble with a firewall; this avoids the use of server ports 1–1023.

If `ssh` connections to a remote site suddenly stop working, it may be the result of tinkering by the sysadmin, most likely a change in `host_key` during system maintenance. After making sure this is the case and nobody is trying to fake the remote host by some clever hack, one can regain connection by removing the `host_key` entry from `$HOME/.ssh/known_hosts` on the local machine.

9.6 Mail programs

Mail configuration divides into three categories:

- MTA: `exim`
- MUA: `mutt`
- Utilities: `procmail`, `fetchmail`, `mail`, ...

9.6.1 Mail transport agent (Exim)

References:

- `exim-doc` and `exim-doc-html` packages
- `http://www.exim.org/`

Use `exim` as the mail transfer agent (MTA). Configure:

```
/etc/exim/exim.conf     "eximconfig" to create and edit
/etc/inetd.conf         comment out smtp to run exim as daemon
/etc/email-addresses    Add spoofed source address lists
check filters using exim -brw, -bf, -bF, -bV, ... etc.
```

A catchall for nonexistent email addresses (Exim)

In `/etc/exim/exim.conf` (Woody or later), in the DIRECTORS part, at the end (after the localuser: director) add a catch-all director that matches all addresses that the previous directors couldn't resolve (per Miquel van Smoorenburg):

```
catchall:
  driver = smartuser
    new_address = webmaster@mydomain.com
```

If one wants to have more a detailed recipe for each virtual domain, etc., add the following at the end of `/etc/exim/exim.conf` (per me, not well tested):

```
*@yourdomain.com ${lookup{$1}lsearch*{/etc/email-addresses} \
        {$value}fail} T
```

Then have an "*" entry in `/etc/email-addresses`.

9.6.2 Mail utility (Fetchmail)

`fetchmail` is run in daemon mode to fetch mail from a POP3 account with an ISP into the local mail system. Configure:

```
/etc/init.d/fetchmail
/etc/rc?.d/???fetchmail run update-rc.d fetchmail default priority 30
/etc/fetchmailrc        configuration file (chown 600, owned by fetchmail)
```

Information on how to start fetchmail as a daemon from the `init.d` script for Potato is confusing (Woody fixed this). See the sample `/etc/init.d/fetchmail` and `/etc/fetchmailrc` files in the example scripts (`examples/`).

If your email headers are contaminated by ^M due to your ISP's mailer, add "stripcr" to your options in `$HOME/.fetchmailrc`:

```
options fetchall no keep stripcr
```

9.6.3 Mail utility (Procmail)

`procmail` is a local mail delivery and filter program. One needs to create `$HOME/.procmailrc` for each account that uses it. Example: _procmailrc (`examples/`)

9.6.4 Mail user agent (Mutt)

Use `mutt` as the mail user agent (MUA) in combination with `vim`. Customize with `~/.muttrc`; for example:

```
# use visual mode and "gq" to reformat quotes
set editor="vim -c 'set tw=72 et ft=mail'"
#
# header weeding taken from the manual (Sven's Draconian header weeding)
#
ignore *
unignore from: date subject to cc
unignore user-agent x-mailer
hdr_order from subject to cc date user-agent x-mailer
auto_view application/msword
....
```

Add the following to `/etc/mailcap` or `$HOME/.mailcap` to display HTML mail and MS Word attachments inline:

```
text/html; lynx -force_html %s; needsterminal;
application/msword; /usr/bin/antiword '%s'; copiousoutput;
description="Microsoft Word Text"; nametemplate=%s.doc
```

9.7 Localization and national language support

Debian is internationalized, offering support for a growing number of languages and local usage conventions. The next subsection lists some of the forms of diversity that Debian currently supports, and the following subsections discuss **localization**, the process of customizing your working environment to allow current input and output of your chosen language(s) and conventions for dates, numeric and monetary formats, and other aspects of a system that differ according to your region.

9.7.1 Language support

Keyboard Debian is distributed with keymaps for nearly two dozen keyboards, and with utilities (in the `kbd` package, or in the `console-tools` package plus the `kbd-compat` package) to install, view, and modify the tables.

Should you need to install a different keyboard, simply run

```
# dpkg-reconfigure console-data
```

and debconf will prompt you for the keymap to use.

Data The vast majority of Debian software packages support handling non-US-ASCII characters through `locale` technology offered by glibc (see 'Locales' on this page).

- 8-bit clean: practically all programs
- other Latin character sets (e.g. ISO-8859-1 or ISO-8859-2): the majority of programs
- multi-byte languages such as Chinese, Japanese or Korean: many new applications

Display X can support all fonts. The list includes not only all the 8-bit fonts but also 16-bit fonts such as Chinese, Japanese or Korean. See 'CJK and X' on page 112.

Translation Translations exist for many of the text messages and documents that are displayed in the Debian system, such as error messages, standard program output, menus, and manual pages. Currently, support for manual pages in German, Spanish, Finnish, French, Hungarian, Italian, Japanese, Korean and Polish is provided through the `manpages-`*`LANG`* packages (where *LANG* is the two-letter ISO country code).

To access an NLS manual page, the user must set the shell LC_MESSAGES variable to the appropriate string. For example, in the case of the Italian-language manual pages, LC_MESSAGES needs to be set to `it`. The `man` program will then search for Italian manual pages under `/usr/share/man/it/`. See 'Locales' on this page for more information on setting locale variables.

9.7.2 Locales

Debian supports **locale** technology. Locale is a mechanism that allows programs to provide suitable output and functionality according to local conventions such as character set, format for date and time, currency symbol, and so on. It uses environment variables to determine the appropriate behavior. For example, assuming you have both the American English and French locales installed on your system, the error messages of many programs can be bilingual:

```
$ LANG="en_US" cat foo
```

```
cat: foo: No such file or directory
$ LANG="fr_FR" cat foo
cat: foo: Aucun fichier ou répertoire de ce type
```

Glibc offers support for this functionality to programs as a library. See `locale(7)`.

9.7.3 Activate locale support capability

Debian does **not** come with all available locales pre-compiled. Check `/usr/lib/locale` to see which locales (besides the default "C") are compiled for your system. If the one you need is not present, you have two options:

- Edit `/etc/locale.gen` to add the desired locale, then run `locale-gen` as root to compile it. See `locale-gen(8)` and the manpages listed in its "SEE ALSO" section.
- Run `dpkg-reconfigure locales` to reconfigure the `locales` package. Or if it is not already installed, installing `locales` will invoke the debconf interface to let you choose needed locales and compile the database.

9.7.4 Activate a particular locale

The following environment variables are evaluated in this order to provide particular locale values to programs:

1. LANGUAGE: This environment variable consists of a colon-separated list of locale names in order of priority. Used only if the POSIX locale is set to a value other than "C" [in Woody; the Potato version always has priority over the POSIX locale]. (GNU extension)

2. LC_ALL: If this is non-null, the value is used for all locale categories. (POSIX.1) Usually "" (null).

3. LC_*: If this is non-null, the value is used for the corresponding category (POSIX.1). Usually "C".

   LC_* variables are:
   - LC_CTYPE: Character classification and case conversion.
   - LC_COLLATE: Collation order.
   - LC_TIME: Date and time formats.
   - LC_NUMERIC: Non-monetary numeric formats.
   - LC_MONETARY: Monetary formats.
   - LC_MESSAGES: Formats of informative and diagnostic messages and interactive responses.
   - LC_PAPER: Paper size.

- LC_NAME: Name formats.
- LC_ADDRESS: Address formats and location information.
- LC_TELEPHONE: Telephone number formats.
- LC_MEASUREMENT: Measurement units (Metric or Other).
- LC_IDENTIFICATION: Metadata about the locale information.

4. LANG: If this is non-null and LC_ALL is undefined, the value is used for all LC_* locale categories with undefined values. (POSIX.1) Usually "C".

Note that some applications (e.g., Netscape 4) ignore LC_* settings.

The `locale` program can display active locale settings and available locales; see `locale(1)`. (NOTE: `locale -a` lists all the locales that your system knows about; this does *not* mean that all of them are compiled! See 'Activate locale support capability' on the page before.)

9.7.5 Beyond `locale`

Programs may need to be configured beyond `locale` configuration to enable a comfortable working environment.

The `language-env` package and its command `set-language-env` greatly eases the configuration of the national language environment on a Debian system.

Also, language-specific entries in the "task" system accessed through `tasksel` or `aptitude` are another useful resource.

For X examples, see 'CJK and X' on page 112.

For more information, see SuSE pages for CJK (`http://www.suse.de/~mfabian/suse-cjk/suse-cjk.html`). Also see the internationalization document, Introduction to i18n (`http://www.debian.org/doc/manuals/intro-i18n/`). It is aimed at developers but is also useful for system administrators.

Chapter 10

Building a gateway with a Debian system

Debian offers an all-purpose gateway machine, which handles NAT, mail, DHCP, DNS cache, HTTP proxy cache, CVS, NFS, and Samba services for a home LAN system. See Netfilter (`http://www.netfilter.org/`), where many network configuration issues are explained.

10.1 Network configuration

10.1.1 Host configuration for the gateway

The LAN uses IP addresses for the following private network range to avoid IP address collision with the Internet.

```
Class A: 10.0.0.0                      with mask 255.0.0.0
Class B: 172.16.0.0 - 172.31.0.0       with mask 255.255.0.0
Class C: 192.168.0.0 - 192.168.255.0 with mask 255.255.255.0
```

Debian uses `/etc/network/interfaces` for IP configuration.

For example, if `eth0` connects to the Internet with a DHCP-provided IP address and `eth1` connects to the LAN, `/etc/network/interfaces` is set as following (Woody or later):

```
auto eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet static
```

```
address 192.168.1.1
network 192.168.1.0
netmask 255.255.255.0
broadcast 192.168.1.255
```

Issue the following command to update the networking configuration to the new `/etc/network/interfaces`:

```
# /etc/init.d/networking restart
```

Reminder: The `interfaces` file in Woody or later releases is not compatible with Potato.

If the system uses a PCMCIA NIC, one needs to set up the network through `/etc/pcmcia/network.opts` instead.

Check the output of the following if in doubt:

```
# ifconfig
# cat /proc/pci
# cat /proc/interrupts
# dmesg|more
```

Sometimes, DSL (PPPoE?) has MTU issues. Refer to the LDP DSL-HOWTO (`http://www.tldp.org/HOWTO/DSL-HOWTO/`). If you have problems with certain websites, see 'Strange access problems with some websites' on page 42.

10.1.2 Network configuration checkpoints

Typical set of programs:

```
# apt-get install nfs samba dhcpd dhcp-client bind squid procmail fetchmail
# apt-get install ssh cvs
```

Then check the following files:

```
/etc/init.d/dhcpd       (edit to serve only LAN = eth1)
/etc/host.allow         (ALL: 192.168.0.0/16 127.0.0.0/8) for NFS
/etc/exports            (Need this for NFS)
/etc/bind/db.192.168.1  (add)
/etc/bind/db.lan        (add)
```

```
/etc/bind/named.conf    (edit)
/etc/resolve.conf       (edit)
/etc/hosts
/etc/dhcpd.conf         (edit for LAN = eth1)
/etc/dhclient.conf      (edit to force local DNS)
/etc/samba/smb.conf
/etc/exim/exim.conf
/etc/mailname
/etc/aliases
/etc/squid.conf         (add all LAN host IPs as allowed)
```

`bind` creates a local cache DNS server and changes DNS to localhost. Check `/etc/resolve.conf`:

```
nameserver 127.0.0.1
search lan.aokiconsulting.com
```

10.2 Netfilter configuration

The netfilter/iptables project is a firewalling subsystem for Linux 2.4 and after. See Netfilter (`http://www.netfilter.org/`), where many network configuration issues are explained.

10.2.1 Basics of netfilter

Netfilter process packets use 5 built-in chains: PREROUTING, INPUT, FORWARD, OUTPUT, and POSTROUTING.

```
                routing
                decision
IN ------> PRE ---> ------> FORWARD -----> ----> POST -----> OUT
interface  ROUTING  \       filter       /       ROUTING     interface
           DNAT     |       tracking     ^       SNAT
           REDIRECT |                    |      MASQUERADE
                    v                    |
                  INPUT                OUTPUT
                    | filter             ^ filter,DNAT
                    v                    |
                    \--> Local Process --/
                            user-space programs
```

10.2.2 Netfilter table

Packets are processed at each built-in chain using the following tables.

- filter (packet filter, default)
  - INPUT (for packets coming into the box itself)
  - FORWARD (for packets being routed through the box)
  - OUTPUT (for locally generated packets).
- nat (network address translation)
  - PREROUTING (for altering packets as soon as they come in)
  - OUTPUT (for altering locally generated packets before routing)
  - POSTROUTING (for altering packets as they are about to go out)
- mangle (network address mangling, good only after 2.4.18)
  - all 5 built-in chains.

10.2.3 Netfilter target

Firewall rules have several targets:

- 4 basic targets:
  - ACCEPT means to let the packet through.
  - DROP means to drop the packet.
  - QUEUE means to pass the packet to userspace (if supported by the kernel).
  - RETURN means stop traversing this chain and resume at the next rule in the previous (calling) chain.
- extended targets:
  - LOG turns on kernel logging.
  - REJECT sends back an error packet and drops the packet.
  - SNAT alters the source address of the packet and is used only in the POSTROUTING chain. (nat table only)
    ```
    --to-source  ipaddr[-ipaddr][:port-port]
    ```
  - MASQUERADE is the same as SNAT but for dynamically assigned IP (dialup) connections. (nat table only)
    ```
    --to-ports port[-port]
    ```
  - DNAT alters that the destination address of the packet and used in the PREROUTING and OUTPUT chains, and user-defined chains which are only called from those chains. (nat table only)
    ```
    --to-destination ipaddr[-ipaddr][:port-port]
    ```
  - REDIRECT alters the destination IP address to send the packet to the machine itself.
    ```
    --to-ports port[-port]
    ```

10.2.4 Netfilter command

The basic commands of `iptables` are:

```
iptables -N chain                    # create a chain

iptables -A chain \                  # add rule to chain
         -t table \                  # use table (filter, nat, mangle)
         -p protocol \               # tcp, udp, icmp, or all,
         -s source-address[/mask] \
         --sport port[:port] \       # source port if -p is tcp or udp
         -d destination-address[/mask] \
         --dport port[:port] \       # dest. port if -p is tcp or udp
         -j target \                 # what to do if match
         -i in-interface-name \      # for INPUT,  FORWARD, PREROUTING
         -o out-interface-name       # for FORWARD, OUTPUT, POSTROUTING
```

10.2.5 IP-masquerade

Machines on the LAN can access Internet resources through a gateway which runs IP-masquerade (NAT) by sharing a single externally accessible IP address.

```
# apt-get install ipmasq
```

Apply example rules to strengthen the `ipmasq` protection. See `/usr/share/doc/ipmasq/examples/stronger/README`. For Debian kernel-image-2.4, make sure to load the proper modules. See 'Network function' on page 76 for the necessary configuration.

For Debian kernel-image-2.2, edit `Z92timeouts.rul` in `/etc/masq/rules` as follows to ensure a longer connection to remote sites (good for large emails, etc.):

```
# tcp, tcp-fin, udp
# 2hr, 10 sec, 160 sec - default
# 1 day, 10 min, 10 min - longer example
$IPCHAINS -M -S 86400 600 600
```

Also, if the network is accessed through a PCMCIA NIC, `ipmasq` needs to be started from `/etc/pcmcia/network.opts`. Read `/usr/share/doc/ipmasq/ipmasq.txt.gz`.

10.2.6 Redirect SMTP connection (2.4)

Suppose you have a notebook PC which is configured to use other LAN environments and you want to use your mail user agent on the notebook PC without reconfiguring it.

Adding the following rules through the `iptables` command to the gateway machine will redirect the SMTP connection to the gateway machine.

```
# iptables -t nat -A PREROUTING -s 192.168.1.0/24 -j REDIRECT \
             -p tcp --dport smtp --to-port 25 # smtp=25, INPUT is open
```

For a more thorough redirect rule set consider installing the `ipmasq` package and adding M30redirect.def (`examples/`) to the `/etc/ipmasq/rules/` directory.

10.3 Manage multiple net connections

[FIXME] Policy routing (by Phil Brutsche `<pbrutsch@tux.creighton.edu>`): See the iproute manual (`http://lartc.org/`) for details. Traffic control (tc) may also be interesting.

Environment:

```
eth0: 192.168.1.2/24; gateway 192.168.1.1
eth1: 10.0.0.2/24; gateway 10.0.0.1
No masquerading on this machine.
```

Special magic:

1. ip rule add from 192.168.1.2 lookup 1
2. ip rule add from 10.0.0.2 lookup 2
3. ip route add to default via 10.0.0.1 metric 0
4. ip route add to default via 192.168.1.1 metric 1
5. ip route add table 1 to 192.168.1.0/24 via eth0
6. ip route add table 1 to 10.0.0.2/24 via eth1
7. ip route add table 1 to default via 192.168.1.1
8. ip route add table 2 to 192.168.1.0/24 via eth0
9. ip route add table 2 to 10.0.0.2/24 via eth1
10. ip route add table 2 to default via 10.0.0.2

[FIXME] I've never done this. How to set up dialup as backup to a fast connection with auto dial-features? Please send me a patch here :)

Chapter 11

Editors

11.1 Popular editors

Linux offers many alternatives for console text editors. Among them:

- `vim`: Powerful and light BSD-heritage editor. VI iMproved.
- `emacs`: Ultimate and heavy GNU-heritage editor. RMS (Richard M. Stallman) original.
- `xemacs`: Emacs: The Next Generation, originally from Lucid.
- `mcedit`: Newbie GNU editor. Identical to `mc` internal editor.
- `ae`: Default small editor (Potato). Avoid this.
- `nano`: Default small GNU editor (Woody). Emulates `pico`.
- `joe`: For WordStar or TurboPascal old-timers.
- `jed`: Fast, full-featured menu-driven editor with Emacs key bindings.
- `jove`: Very small editor with Emacs key bindings.
- `nvi`: New vi. Bug-for-bug compatible with the original vi.

Use `update-alternatives --config editor` to change the default editor.

Also a few X-based text editors are noteworthy:

- `gvim`: Vim with GUI (`vim-gtk` package)
- `emacs`: The One True Emacs (auto-detect X).
- `xemacs`: Next generation Emacs (auto detect X).

These xclient commands take standard options such as `-fn` *`a24`*, which makes life easy for older folks like me :) See 'X client' on page 108.

11.2 Rescue editors

There are a few editors which reside in `/bin`. One of these should be installed to ease editing files when `/usr` is not accessible.

- `elvis-tiny`: Minimum vi editor (`vi` to start)
- `nano-tiny`: Minimum non-vi editor (`nano-tiny` to start)
- `ed`: Minimum editor (always there but tough to use)

11.3 Emacs and Vim

11.3.1 Vim hints

Read the “VIM - main help file” document by pressing **<F1>** while running the program.

```
<F1>           Help
<esc>          Back to normal mode
V              Visual mode
i              Insert mode
:              Command-line commands
:set tw=72     Set text width to 72
<F11>          Insert (paste) mode
:r! date -R    Insert RFC-822 date
qa             Record keystrokes into register a
@a             Execute keystrokes from register a
:edit foo.txt  Edit another file by loading foo.txt
:wnext         Write current file and edit next file
```

`q` and `@` can be used for simple macro recording and playback. For instance, to create a macro to that inserts HTML italics tags around the word at the cursor, you could enter `qii<i>^[ea</i>^[q` (where `^[` is the ESC key). Then typing `@i` at the start of a word would add the tags **<i>** and **</i>**.

11.3.2 Emacs hints

```
<F1>                Help
<F10>               Menu
C-u M-! date -R     Insert RFC-822 date
```

11.3.3 Starting the editor

```
start editor:                  emacs filename  vim filename
start in vi compatible:                        vim -C
start in vi non-compatible:                    vim -N
start with compile default:    emacs -q        vim -N -u NONE
```

11.3.4 Editor command summary (Emacs, Vim)

```
exit:                           C-x C-c         :qa /:wq /:xa /:q!
Get back/command mode:          C-g             <esc>
Backward(left):                 C-b             h
Forward(right):                 C-f             l
Next(down):                     C-n             j
Previous(up):                   C-p             k
stArt of line(^):               C-a             0
End of line($):                 C-e             $
mUltiple commands:              C-u nnn cmd     :count cmd
Multiple commands:              M-digitkey cmd
save File:                      C-x C-f         :w file
beginning of buffer:            M-<             1G
end of buffer:                  M->             G
scroll forward 1 screen:        C-v             ^F
scroll forward 1/2 screen:                      ^D
scroll forward 1 line:                          ^E
scroll backward 1 screen:       M-v             ^B
scroll backward 1/2 screen:                     ^U
scroll backward 1 line:                         ^Y
scroll the other window:        M-C-v
delete under cursor:            C-d             x
delete from cursor to eol:      C-k             D
iSearch forward:                C-s
isearch Reverse:                C-r
Search forward:                 C-s enter       /
search Reverse:                 C-r enter       ?
isearch regexp:                 M-C-s
isearch backward regexp:        M-x isearch-backward-regexp
search regexp:                  M-C-s enter     /
search backward regexp:         M-x isearch-backward-regexp enter
                                                ?
Help:                           C-h C-h         :help
Help Apropos:                   C-h a
Help key Bindings:              C-h b           :help [key]
Help Info:                      C-h i
Help Major mode:                C-h m
Help tutorial:                  C-h t           :help howto
Undo:                           C-_             u
Redo:                           C-f             ^R
Mark cursor position:           C-@             m{a-zA-Z}
```

```
eXchange Mark and position:     C-x C-x
goto mark in current file:                      '{a-z}
goto mark in any file:                          '{A-Z}
copy region:                    M-w             {visual}y
kill region:                    C-w             {visual}d
Yank and keep buffer:           C-y
Yank from kill buffer:          M-y             p
convert region to Upper:        C-x C-u         {visual}U
convert region to Lower:        C-x C-l         {visual}u
Insert special char:            C-q octalnum/keystroke
                                                ^V decimal/keystroke
replace:                        M-x replace-string      :%s/aaa/bbb/g
replace regexp:                 M-x replace-regexp      :%s/aaa/bbb/g
query replace:                  M-%                     :%s/aaa/bbb/gc
query replace:                  M-x query-replace
query replace regexp:           M-x query-replace-regexp
Open file:                      C-x C-f         :r file
Save file:                      C-x C-s         :w
Save all buffers:               C-x s           :wa
Save as:                        C-x C-w file    :w file
Prompt for buffer:              C-x b
List buffers:                   C-x C-b         :buffers
Toggle read-only:               C-x C-q         :set ro
Prompt and kill buffer:         C-x k
Split vertical:                 C-x 2           :split
Split horizontal:               C-x 3           :vsplit (ver. 6)
Move to other window:           C-x o           ^Wp
Delete this window:             C-x 0           :q
Delete other window(s):         C-x 1           ^Wo
run shell in bg:                M-x compile
kill shell run in bg:           M-x kill-compilation
run make                                        :make Makefile
check error message:            C-x`            :echo errmsg
run shell and record:           M-x shell       :!script -a tmp
...clean BS, ...                                :!col -b <tmp >record
...save/recall shell record:    C-x C-w record  :r record
run shell:                      M-! sh          :sh
run command:                    M-! cmd         :!cmd
run command and insert:         C-u M-! cmd     :r!cmd
run filter:                     M-| file        {visual}:w file
run filter and insert:          C-u M-| filter  {visual}:!filter
show option                                     :se[t] {option}?
```

`reset option to default`	`:se[t] {option}&`
`reset boolean option`	`:se[t] no{option}`
`toggle boolean option`	`:se[t] inv{option}`
`wrap text at column 72`	`:se tw=72`
`do not wrap`	`:se tw=0`
`autoindent`	`:se ai`
`expand tab`	`:se et`
`specify comment (mail)`	`:se comments=n:>,n:\`

`run GDB`	`M-x gdb`
`describe GDB mode`	`C-h m`
`step one line`	`M-s`
`next line`	`M-n`
`step one instruction (stepi)`	`M-i`
`finish current stack frame`	`C-c C-f`
`continue`	`M-c`
`up arg frames`	`M-u`
`down arg frames`	`M-d`
`copy number from point, insert at the end`	`C-x &`
`set break point`	`C-x SPC`

11.3.5 Vim configuration

In order to use all Vim features and syntax highlighting, include the following lines in `~/.vimrc` or `/etc/vimrc`:

```
set nocompatible
set nopaste
set pastetoggle=<f11>
syn on
```

Paste mode enables one to avoid autoindent interfering with cut-and-paste operations on a console terminal. It does more than just a simple ":set noai".

See 'Using GnuPG with Vim' on page 163 for GnuPG integration.

11.3.6 Ctags

`apt-get install exuberant-ctags` and run ctags on the source files. Type `:tag` *`function_name`* in Vim to jump to the line where *function_name* starts. The tags work for C, C++, Java, Python, and many other languages.

Emacs has the same ctags capabilities.

11.3.7 Convert a syntax-highlighted screen to HTML source

`so \$VIMRUNTIME/syntax/2html.vim` from Vim command mode will convert highlighted text to HTML text. Save with `:w file.html` and `:q`. Useful for C code, etc.

11.3.8 Split screen with `vim`

`vim` can edit multiple files in a multi-split-screen environment. Type `:help usr_08.txt` for details.

To split the screen display between different files, type at the vi command prompt:

```
:split another-file
:vsplit another-file
```

Or at a shell prompt:

```
$ vi -o file1.txt file2.txt   # Horizontal split
$ vi -O file1.txt file2.txt   # Vertical split
```

will provide multi-window vi.

```
$ vimdiff file.txt~ file.txt         # check recent changes of file.txt
$ vimdiff file.en.sgml file.fr.sgml  # check changes of translation
$ gvimdiff file.txt~ file.txt        # in X
```

will provide a nice view of differences between an original and a backup file. In SGML it matches tags, so comparing translations in this mode works very well.

Special cursor movements with CTRL-W commands:

```
CTRL-W +      increase the size of a window
CTRL-W -      decrease the size of a window
CTRL-W h      move to the window left
CTRL-W j      move to the window below
CTRL-W k      move to the window above
CTRL-W l      move to the window right
...
```

Use the following to control screen scrolling:

```
:set scrollbind
:set noscrollbind
```

Chapter 12

CVS

Check `/usr/share/doc/cvs/html-cvsclient`,`/usr/share/doc/cvs/html-info`,`/usr/share/doc/cvsbook` with lynx or run `info cvs` and `man cvs` for detailed information.

12.1 Installing a CVS server

The following setup will allow commits to the CVS repository only by a member of the "src" group, and administration of CVS only by a member of the "staff" group, thus reducing the chance of shooting oneself.

```
# cd /var/lib; umask 002 ; sudo mkdir cvs # [Woody] FSH
# apt-get install cvs cvs-doc cvsbook
# export CVSROOT=/var/lib/cvs
# cd $CVSROOT
# chown root:src .  # "staff" to restrict more for starting project.
# chmod 3775 .             # If above uses "staff", use 2775
# cvs -d /var/lib/cvs init # safer to specify -d here explicitly!
# cd CVSROOT
# chown -R root:staff .
# chmod 2775 .
# touch val-tags
# chmod 664 history val-tags
# chown root:src history val-tags
```

12.2 CVS session examples

The following will set up shell environments for the CVS repository access.

12.2.1 Anonymous CVS (download only)

Read-only remote access:

```
$ export CVSROOT=:pserver:anonymous@cvs.qref.sf.net:/cvsroot/qref
$ cvs login
$ cvs -z3 co qref
```

12.2.2 Use local CVS server

Local access from a shell on the same machine:

```
$ export CVSROOT=/var/lib/cvs
```

12.2.3 Use remote CVS pserver

Remote access without SSH (use RSH protocol capability in `cvs`):

```
$ export CVSROOT=:pserver:account@cvs.foobar.com:/var/lib/cvs
$ cvs login
```

This is prone to eavesdropping attack.

12.2.4 Use remote CVS through `ssh`

Remote access with SSH:

```
$ export CVSROOT=:ext:account@cvs.foobar.com:/var/lib/cvs
```

or for Sourceforge:

```
$ export CVSROOT=:ext:account@cvs.qref.sf.net:/cvsroot/qref
```

You can also use RSA authentication ('Connect with fewer passwords' on page 115), which eliminates the password prompt.

12.2.5 Create a new CVS archive

For,

ITEM	VALUE	MEANING
source tree:	*~/project-x*	All source codes
Project name:	*project-x*	Name for this project
Vendor Tag:	*Main-branch*	Tag for the entire branch
Release Tag:	*Release-original*	Tag for a specific release

Then,

```
$ cd ~/project-x                 # dive into source directory
 ... create a source tree ...
$ cvs import -m "Start project-x" project-x Main-branch Release-initial
$ cd ..; rm -R ~/project-x
```

12.2.6 Work with CVS

To work with *project-x* using the local CVS repository:

```
$ cd                            # move to the work area
$ cvs co project-x              # get sources from CVS to local
$ cd project-x
 ... make changes to the content ...
$ cvs diff -u                   # similar to diff -u repository/ local/
$ cvs ci -m "Describe change"   # save local sources to CVS
$ vi newfile_added
$ cvs add newfile_added
$ cvs ci -m "Added newfile_added"
$ cvs up                        # merge latest version from CVS
 ... watch out for lines starting with "C filename"
 ... unmodified code is moved to `.#filename.version'.
 ... Search "<<<<<<<" and ">>>>>>>" in filename.
$ cvs tag Release-1             # add release tag
 ... edit further ...
$ cvs tag -d Release-1          # remove release tag
$ cvs ci -m "more comments"
$ cvs tag Release-1             # re-add release tag
$ cd                            # move back to the work area
```

```
$ cvs co -r Release-initial -d old project-x
 ... get original version to old directory
$ cd old
$ cvs tag -b Release-initial-bugfixes # create branch (-b) tag
 ... Now you can work on the old version (Tag=sticky)
$ cvs update
 ... Source tree now has sticky tag "Release-initial-bugfixes"
 ... Work on this branch
$ cvs up # sync with files modified by others on this branch
$ cvs ci -m "check into this branch"
$ cvs update -kk -A
 ... Remove sticky tag and forget contents
 ... Update from main trunk without keyword expansion
$ cvs update -kk -j Release-initial-bugfixes
 ... Merge from Release-initial-bugfixes branch into the main
 ... trunk without keyword expansion.  Fix conflicts with editor.
$ cvs ci -m "merge Release-initial-bugfixes"
$ cd
$ tar -cvzf old-project-x.tar.gz old     # make archive, -j for bz2
$ cvs release -d old               # remove local source (optional)
```

Nice options to remember (use as first argument(s) to `cvs`):

```
-n      dry run, no effect
-t      display messages showing steps of cvs activity
```

12.2.7 Export files from CVS

To get the latest version from CVS, use “tomorrow”:

```
$ cvs ex -D tomorrow module_name
```

12.2.8 Administer CVS

Add alias to a project (local server):

```
$ su - admin           # a member of staff
$ export CVSROOT=/var/lib/cvs
$ cvs co CVSROOT/modules
```

```
$ cd CVSROOT
$ echo "px -a project-x" >>modules
$ cvs ci -m "Now px is an alias for project-x"
$ cvs release -d .
$ exit                 # control-D to get back from su
$ cvs co -d project px
 ... check out project-x (alias:px) from CVS to directory project
$ cd project
 ... make changes to the content ...
```

12.3 Troubleshooting CVS

12.3.1 File permissions in repository

CVS will not overwrite the current repository file but replaces it with another one. Thus, *write permission to the repository directory* is critical. For every new repository creation, run the following to ensure this condition if needed.

```
# cd /var/lib/cvs
# chown -R root:src repository
# chmod -R ug+rwX   repository
# chmod    2775     repository  # if needed, this and subdirectory
```

12.3.2 Execution bit

A file's execution bit is retained when checked out. Whenever you see execution permission problems in checked-out files, change permissions of the file in the CVS repository with the following command.

```
# chmod ugo-x filename
```

12.4 CVS commands

Here are CVS commands with their shortcuts.

```
{add|ad|new} [-k kflag] [-m 'message'] files...
{admin|adm|rcs} [rcs-options] files...
```

```
{annotate|ann} [options] [files...]
{checkout|co|get} [options] modules...
{commit|ci|com}   [-lnR]  [-m  'log_message'  |  -f  file] \
        [-r revision] [files...]
{diff|di|dif} [-kl] [rcsdiff_options] [[-r rev1 | -D date1] \
        [-r rev2 |  -D date2]] [files...]
{export|ex|exp} [-flNn] -r rev|-D date [-d dir] [-k kflag] module...
{history|hi|his} [-report] [-flags] [-options args] [files...]
{import|im|imp} [-options] repository vendortag releasetag...
{login|logon|lgn}
{log|lo|rlog} [-l] rlog-options [files...]
{rdiff|patch|pa} [-flags] [-V vn] [-r t|-D d [-r t2|-D d2]] modules...
{release|re|rel} [-d] directories...
{remove|rm|delete} [-lR] [files...]
{rtag|rt|rfreeze} [-falnR]  [-b]  [-d]  [-r  tag  |  -D  date] \
         symbolic_tag modules...
{status|st|stat} [-lR] [-v] [files...]
{tag|ta|freeze} [-lR] [-F] [-b] [-d] [-r tag | -D date]  [-f] \
         symbolic_tag [files...]
{update|up|upd} [-AdflPpR] [-d] [-r tag|-D date] files...
```

Chapter 13

Programming

Do not use "test" as the name of an executable test file. `test` is a shell builtin.

13.1 Where to start

References:

- Documents and examples under `/usr/share/doc/`*`packages`*
- Unix / Programming Information (`http://arioch.unomaha.edu/~jclark/#info`)
- *Linux Programming Bible* (John Goerzen/IDG books)

Many long info documents can be obtained as paperbacks from GNU (`http://www.gnu.org/`).

The next four sections contain sample scripts in different languages for creating a text file of account information to be added to `/etc/passwd` using a batch processor such as the `newusers` program. Each script requires as input a file with lines of the form `firstname lastname password`. (Actual user home directories will not be created via these scripts.)

13.2 Shell

Reading shell scripts is the **best** way to understand how a Unix-like system works. Here, I give some pointers and reminders for shell programming.

13.2.1 Bash – GNU standard interactive shell

References for Bash:

- `bash(1)`

- `info bash`
- the LDP BASH Programming - Introduction HOWTO (http://www.tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html) as starter information.
- `mc /usr/share/doc/bash/examples/ /usr/share/doc/bash/` (Install the `bash-doc` package to see the example files.)
- *Learning the bash Shell*, 2nd edition (O'Reilly)

Short program example (create account entries for `newusers` from standard input):

```
#!/bin/bash
# (C) Osamu Aoki Sun Aug 26 16:53:55 UTC 2001 Public Domain
pid=1000;
while read n1 n2 n3 ; do
if [ ${n1:0:1} != "#" ]; then
let pid=$pid+1
echo ${n1}_${n2}:password:${pid}:${pid}:,,,/home/${n1}_${n2}:/bin/bash
fi
done
```

13.2.2 POSIX shells

Several packages provide a POSIX shell in Debian:

- `ash`
  - Priority: optional
  - Installed-Size: 180
  - Smaller and much faster — good for initial boot
- `bash`
  - Essential: yes
  - Priority: required
  - Installed-Size: 580
  - Larger and featureful — many extensions implemented
- `pdksh`
  - Priority: optional
  - Installed-Size: 408
  - Complete AT&T ksh look-alike

If you are writing shell script for portability, it is best to write it as a POSIX shell script. Use `/bin/sh` linked to `ash` to test its POSIX compliance. Avoid writing scripts with a **bashism** such as

```
#! /bin/bash
export ENVPARM=value
```

Instead, make it POSIX-compliant:

```
#!/bin/sh
ENVPARM=value
export ENVPARM
```

13.2.3 Shell parameters

Several **special parameters** to remember:

```
$0      = name of the shell or shell script
$1      = first(1) shell argument
 ...
$9      = ninth(9) shell argument
$#      = number of positional parameters
"$*"    = "$1 $2 $3 $4 ... $n"
"$@"    = "$1" "$2" "$3" "$4" ... "$n"
$?      = exit status of the most recent command
$$      = PID of this shell script
$!      = PID of most recently started background job
```

Basic **parameter expansions** to remember:

```
    Form        If var is set(*)    If var is not set(*)
${var:-string}  $var                string
${var:+string}  string              null
${var:=string}  $var                string
                                    (and run var=string)
${var:?string}  $var                (echo string and then exit)
```

Here, the colon ':' in all of these operators is actually optional.

- With ':' = operator test for "exist" and "not null".
- Without ':' = operator test for "exist" only.

Basic **parameter substitutions** to remember:

```
    Form        Result
${var%suffix}   Remove smallest suffix pattern
${var%%suffix}  Remove largest suffix pattern
${var#prefix}   Remove smallest prefix pattern
${var##prefix}  Remove largest prefix pattern
```

13.2.4 Shell redirection

Basic **redirection** to remember (here the [n] is an optional number):

`[n]>` *file*	Redirect standard output (or n) to *file*.
`[n]>>` *file*	Append standard output (or n) to *file*.
`[n]<` *file*	Redirect standard input (or n) from *file*.
`[n1]>&n2`	Redirect standard output (or n1) to n2.
`>` *file* `>&2`	Redirect standard and error output to *file*.
`|` *command*	Pipe standard output (or n) to *command*.
`>&2 |` *command*	Pipe standard and error output to *command*.

13.2.5 Shell conditionals

Each command returns an **exit status** which can be used for the conditional expression:

- Success: 0 (True)
- Error: 1 - 255 (False)

Note that the use here of a 0 value to mean "true" differs from the usual convention in some other areas of computing. Also, '[' is the equivalent of the `test` command, which evaluates its arguments up to ']' as a conditional expression.

Basic **conditional idioms** to remember are:

```
command && if_success_run_this_command_too
command || if_not_success_run_this_command_too

if [ conditional_expression ]; then
 if_success_run_this_command
else
 if_not_success_run_this_command
fi
```

File comparison operators in the conditional expression are:

`-e` *file*	True if *file* exists.
`-d` *file*	True if *file* exists and is a directory.
`-f` *file*	True if *file* exists and is a regular file.
`-w` *file*	True if *file* exists and is writable.
`-x` *file*	True if *file* exists and is executable.
file1 `-nt` *file2*	True if *file1* is newer than *file2*. (modification)
file1 `-ot` *file2*	True if *file1* is older than *file2*. (modification)
file1 `-ef` *file2*	True if they are the same device and inode numbers.

String comparison operators in the conditional expression are:

`-z` *str*	True if the length of *str* is zero.
`-n` *str*	True if the length of *str* is non-zero.

```
str1 == str2   True if the str are equal.
str1 = str2    True if the str are equal.
               ( = may be used in place of == )
str1 != str2   True if the str are not equal.
str1 <  str2   True if str1 sorts before str2 (locale dependent).
str1 >  str2   True if str1 sorts after str2 (locale dependent).
```

Arithmetic integer comparison operators in the conditional expression are `-eq`, `-ne`, `-lt`, `-le`, `-gt`, or `-ge`.

13.2.6 Command-line processing

The shell processes a script as follows:

- split into **tokens** by the metacharacters: SPACE, TAB, NEWLINE, ;, (,), <, >, |, &
- check **keyword** if not within “…” or ’…’ (loop)
- expand **alias** if not within “…” or ’…’ (loop)
- expand **brace**, A{b|c} -> Ab Ac, if not within “…” or ’…’
- expand **tilde**, ~*user* -> $HOME/$USER, if not within “…” or ’…’
- expand **parameter**, $*PARAMETER*, if not within ’…’
- expand **command substitution**, $(*command*), if not within ’…’
- split into **words** with $IFS if not within “…” or ’…’
- expand **pathname** *?[] if not within “…” or ’…’
- lookup **command**
  - function
  - built-in
  - file in $PATH
- loop

Single quotes within double quotes have no effect.

13.3 Awk

References for Awk:

- *Effective awk Programming*, 3rd edition (O’Reilly)
- *Sed & awk*, 2nd edition (O’Reilly)
- `mawk(1)` and `gawk(1)`
- `info gawk`

Short program example (create `newusers` command entry):

```
#!/usr/bin/awk -f
```

```
# Script to create a file suitable for use in the 'newusers' command,
# from a file consisting of user IDs and passwords in the form:
# First Last password
# Copyright (c) KMSelf Sat Aug 25 20:47:38 PDT 2001
# Distributed under GNU GPL v 2, or at your option, any later version.
# This program is distributed WITHOUT ANY WARRANTY.

BEGIN {
    # Assign starting UID, GID
    if ( ARGC > 2 ) {
 startuid = ARGV[1]
 delete ARGV[1]
    }
    else {
        printf( "Usage:  newusers startUID file\n" \
   "...where startUID is the starting userid " \
   "to add, and file is \n" \
   "an input file in form firstname last name password\n" \
 )
 exit
    }

    infile = ARGV[1]
    printf( "Starting UID: %s\n\n", startuid )
}

/^#/ { next }

{
    ++record
    first = $1
    last = $2
    passwd = $3
    user= substr( tolower( first ), 1, 1 ) tolower( last )
    uid = startuid + record - 1
    gid = uid
    printf( "%s:%s:%d:%d:%s %s,,/home/%s:/bin/bash\n",  \
 user, passwd, uid, gid, first, last, user \
 )
}
```

Two packages provide POSIX `awk` in Debian:

- `mawk`
  - Priority: required
  - Installed-Size: 228
  - Smaller and much faster — good for default install
  - Compile-time limits exist
    - NF = 32767
    - sprintf buffer = 1020
- `gawk`
  - Priority: optional
  - Installed-Size: 1708
  - Larger and featureful — many extensions implemented
    - System V Release 4 version of UNIX
    - Bell Labs awk
    - GNU-specific

13.4 Perl

This is **the** interpreter on a Unix-like system.

References for Perl:

- `perl(1)`
- *Programming Perl*, 3rd edition (O'Reilly)

Short program example (create `newusers` command entry):

```
#!/usr/bin/perl
# (C) Osamu Aoki Sun Aug 26 16:53:55 UTC 2001 Public Domain
$pid=1000;
while (<STDIN>) {
 if (/^#/) { next;}
        chop;
        $pid++;
        ($n1, $n2, $n3) = split / /;
        print $n1,"_",$n2,":", $n3, ":",$pid,
                  ":",$pid,",,,/home/",$n1,"_",$n2,":/bin/bash\n"
}
```

Install Perl module *module name*:

```
# perl -MCPAN -e 'install modulename'
```

13.5 Python

It’s a nice object-oriented interpreter.

References for Python:

- `python(1)`
- *Learning Python* (O’Reilly).

Short program example (create `newusers` command entry):

```
#! /usr/bin/env python
import sys, string

# (C) Osamu Aoki Sun Aug 26 16:53:55 UTC 2001 Public Domain
# Ported from awk script by KMSelf Sat Aug 25 20:47:38 PDT 2001
# This program is distributed WITHOUT ANY WARRANTY.

def usages():
    print \
"Usage:  ", sys.argv[0], " start_UID [filename]\n" \
"\tstartUID is the starting userid to add.\n" \
"\tfilename is input file name. If not specified, standard input.\n\n" \
"Input file format:\n"\
"\tfirstname lastname password\n"
 return 1

def parsefile(startuid):
    #
    # main filtering
    #
    uid = startuid
    while 1:
        line = infile.readline()
        if not line:
            break
        if line[0] == ’#’:
            continue
        (first, last, passwd) = string.split(string.lower(line))
        # above crashes with wrong # of parameters :-)
        user = first[0] + last
        gid = uid
        lineout = "%s:%s:%d:%d:%s %s,,/home/%s:/bin/bash\n" %  \
            (user, passwd, uid, gid, first, last, user)
```

```
        sys.stdout.write(lineout)
        +uid

if __name__ == ’__main__’:
    if len(sys.argv) == 1:
        usages()
    else:
        uid = int(sys.argv[1])
        #print "# UID start from: %d\n" % uid
        if len(sys.argv) > 1:
            infilename   = string.join(sys.argv[2:])
            infile = open(infilename, ’r’)
            #print "# Read file from: %s\n\n" % infilename
        else:
            infile = sys.stdin
        parsefile(uid)
```

13.6 Make

References for Make:

- `info make`
- `make(1)`
- *Managing Projects with make*, 2nd edition (O’Reilly)

Simple automatic variables:

Rule syntax:

```
Target: [ Prerequisite ... ]
 [TAB]  command1
 [TAB]  -command2 # ignore errors
 [TAB]  @command3 # suppress echoing
```

Here `[TAB]` is a TAB code. Each line is interpreted by the shell after make variable substitution. Use `\` at the end of a line to continue the script. Use `$$` to enter `$` for environment values for a shell script.

Implicit rule equivalents:

```
.c:   header.h == %  : %.c header.h
.o.c: header.h == %.c: %.o header.h
```

Automatic variables for above rules:

```
foo.o: new1.c new2.c.c old1.c new3.c
$@ == foo.o                         (target)
$< == new1.c                        (first one)
$? == new1.c new2.c new3.c          (newer ones)
$^ == new1.c new2.c.c old1.c new3.c (all)
$* == `%' matched stem in the target pattern.
```

Variable references:

```
foo1 := bar    # One-time expansion
foo2  = bar    # Recursive expansion
foo3 += bar    # Append
SRCS := $(wildcard *.c)
OBJS := $(foo:c=o)
OBJS := $(foo:%.c=%.o)
OBJS := $(patsubst %.c,%.o,$(foo))
DIRS  = $(dir directory/filename.ext) # Extracts "directory"
$(notdir NAMES...), $(basename NAMES...), $(suffix NAMES...) ...
```

Run `make -p -f/dev/null` to see automatic internal rules.

13.7 C

Preparation:

```
# apt-get install glibc-doc manpages-dev libc6-dev gcc
```

References for C:

- `info libc` (C library function reference)
- `gcc(1)`
- `<var>each_C_library_function_name</var>(3)`
- Kernighan & Ritchie, *The C Programming Language*, 2nd edition (Prentice Hall).

13.7.1 Simple C program (gcc)

A simple example to compile `example.c` with a library `libm` into an executable `run_example`:

```
$ cat > example.c
#include <stdio.h>
#include <math.h>
#include <string.h>

int main(int argc, char **argv, char **envp){
        double x;
        char y[11];
        x=sqrt(argc+7.5);
        strncpy(y, argv[0], 10); /* prevent buffer overflow */
        y[10] = '\0'; /* fill to make sure string ends with '\0' */
        printf("%5i, %5.3f, %10s, %10s\n", argc, x, y, argv[1]);
        return 0;
}

$ gcc -Wall -g -o run_example example.c -lm
$ ./run_example
        1, 2.915, ./run_exam,     (null)
$ ./run_example 1234567890qwerty
        2, 3.082, ./run_exam, 1234567890qwerty
```

Here, `-lm` is needed to link library `libm` for sqrt(). The actual library is in `/lib` with filename `libm.so.6`, which is a symlink to `libm-2.1.3.so`.

Look at the last parameter in the output text. There are more than 10 characters even though `%10s` is specified.

The use of pointer memory operation functions without boundary checks, such as `sprintf` and `strcpy`, is deprecated to prevent buffer overflow exploits that leverage the above overrun effects. Instead, use `snprintf` and `strncpy`.

13.7.2 Debugging

Debugging with gdb

Preparation:

```
# apt-get install gdb
```

References for `gdb`:

- `info gdb` (tutorial)

- `gdb(1)`

Use `gdb` to debug a program compiled with the `-g` option. Many commands can be abbreviated. Tab expansion works as in the shell.

```
$ gdb program
(gdb) b 1                # set breakpoint at line 1
(gdb) run arg1 arg2 arg3 # run program
(gdb) next               # next line
...
(gdb) step               # step forward
...
(gdb) p parm             # print parm
...
(gdb) p parm=12          # set value to 12
```

For debugging from within Emacs, refer to 'Editor command summary (Emacs, Vim)' on page 131.

Check dependency on libraries

Use `ldd` to find out a program's dependency on libraries:

```
$ ldd /bin/ls
        librt.so.1 => /lib/librt.so.1 (0x4001e000)
        libc.so.6 => /lib/libc.so.6 (0x40030000)
        libpthread.so.0 => /lib/libpthread.so.0 (0x40153000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

For `ls` to work in a `chroot`ed environment, the above libraries must be available in your `chroot`ed environment.

The following commands will also be useful:

- `strace`: trace system calls and signals
- `ltrace`: trace library calls

Debugging with memory leak detection tools

There are several memory leak detection tools available in Debian.

- `njamd`
- `valgrind`

- `dmalloc`
- `electric-fence`
- `memprof`
- `memwatch`
- `mpatrol`
- `leaktracer`
- `libgc6`
- `Insure++` from Parasoft (`http://www.parasoft.com`). (non-free, commercial for fee)

Also check out Debugging Tools for Dynamic Storage Allocation and Memory Management (`http://www.cs.colorado.edu/homes/zorn/public_html/MallocDebug.html`).

13.7.3 Flex – a better Lex

`flex` is a fast lexical analyzer generator.

References for `flex`:
- `info flex` (tutorial)
- `flex(1)`

You need to provide your own `main()` and `yywrap()`, or your `program.l` should look like this to compile without a library (`yywrap` is a macro; `%option main` turns on `%option noyywrap` implicitly):

```
%option main
%%
.|\n ECHO ;
%%
```

Alternatively, you may compile with the `-lfl` linker option at the end of your `cc` command line (like ATT-Lex with `-ll`). No `%option` is needed in this case.

13.7.4 Bison – a better Yacc

A few packages provide a Yacc-compatible LALR parser generator in Debian:
- `bison`: GNU LALR parser generator
- `byacc`: The Berkeley LALR parser generator
- `byyacc`: Backtracking parser generator based on byacc

References for `bison`:
- `info bison` (tutorial)
- `bison(1)`

You need to provide your own `main()` and `yyerror()`. `main()` calls `yyparse()` which calls `yylex()`, usually created with FleX.

```
%%

%%
```

13.7.5 Autoconf

`autoconf` is a tool for producing shell scripts that automatically configure software source code packages to adapt to many kinds of UNIX-like systems using the entire GNU build system.

`autoconf` produces the configuration script `configure`. `configure` automatically creates a customized `Makefile` and `Makefile.am`.

Compile and install a program

Debian does not touch files in `/usr/local` (see 'Supporting diversity' on page 23). So if you compile a program from source, install it into `/usr/local` so it will not interfere with Debian.

```
$ cd src
$ ./configure --prefix=/usr/local
$ make
$ make install # this puts the files in the system
```

Uninstall program

If you still have the source and if it uses `autoconf/automake` and if you can remember how you configured it:

```
$ ./configure all-of-the-options-you-gave-it
# make uninstall
```

Alternatively, if you are absolutely sure there is nothing important in `/usr/local`, you can erase all its contents by:

```
# find /usr/local -type f -print0 | xargs -0 rm -f
```

13.8 Document preparation

13.8.1 `roff` typesetting

Traditionally, `roff` is the main Unix text processing system.

See `roff(7)`,`groff(7)`,`groff(1)`,`grotty(1)`,`troff(1)`,`groff_mdoc(7)`,`groff_man(7)`, `groff_ms(7)`,`groff_me(7)`,`groff_mm(7)`,and "`info groff`".

A good tutorial on `-me` macros exists. If you have groff (1.18 or newer), find `/usr/share/doc/groff/meintro.me.gz` and do the following:

```
$ zcat /usr/share/doc/groff/meintro.me.gz | \
     groff -Tascii -me - | less -R
```

The following will make a completely plain text file:

```
$ zcat /usr/share/doc/groff/meintro.me.gz | \
    GROFF_NO_SGR=1 groff -Tascii -me - | col -b -x > meintro.txt
```

For printing, use PostScript output.

13.8.2 SGML

Preparation:

```
# apt-get install debiandoc-sgml debiandoc-sgml-doc
```

References for `debiandoc-sgml`:

- `/usr/share/doc/debiandoc-sgml-doc`
- `debiandoc-sgml(1)`
- *DocBook: The Definitive Guide*, by Walsh and Muellner (O'Reilly)

SGML enables management of multiple formats of a document. One easy SGML system is Debiandoc, which is used here. This requires minor conversion from original text files for the following characters:

```
<   &lt;
>   &gt;
"~" "&nbsp;"  (nonbreakable space)
```

```
&    &amp;
©    &copy;
-    &ndash;
--    &mdash;
```

To mark a section as a nonprintable comment, enter:

```
<!-- State issue here ... -->
```

To mark a section with a switchable comment, enter:

```
<![ %FIXME [ State issue here ... ]]>
```

In SGML, the *first definition* of an entity wins. For example:

```
<!entity % qref "INCLUDE">
<![ %qref [ <!entity param "Data 1"> ]]>
<!entity param "Data 2">
&param;
```

This ends up as “Data 1”. If the first line has “IGNORE” instead of “INCLUDE”, this ends up as “Data 2” (the second line is a conditional statement). Also, repeating phrases can be defined in advance separately from the context.

```
<!entity whoisthis "my">
Hello &whoisthis; friend.
This is &whoisthis; book.
```

This results in the following:

```
Hello my friend.
This is my book.
```

See the short SGML example `sample.sgml` in the examples (`examples/`).

When SGML documents become bigger, sometimes TeX may cause errors. You must increase pool size in `/etc/texmf/texmf.cnf` (or more appropriately edit `/etc/texmf/texmf.d/95NonPath` and run `update-texmf`) to fix this.

13.9 Packaging

Preparation:

```
# apt-get install debian-policy developers-reference \
        maint-guide dh-make debhelper
# apt-get install packaging-manual # if Potato
```

References for packaging:
- 'The Debian package management system' on page 11 (basics)
- Debian New Maintainers' Guide (tutorial)
- `dh-make(1)`
- Debian Developer's Reference (best practice)
- Debian Policy Manual (authoritative)
- Packaging Manual (Potato)

Use `dh_make` from the `dh-make` package to create a baseline package. Then, proceed according to instructions in `dh-make(1)`. This uses `debhelper` in `debian/rules`.

An older approach is to use `deb-make` from the `debmake` package. This uses no `debhelper` scripts and depends only on the shell.

For examples of multiple-source packages, see "mc" (`dpkg-source -x mc_4.5.54.dsc`), which uses "sys-build.mk" by Adam Heath (`<doogie@debian.org>`), and "glibc" (`dpkg-source -x glibc_2.2.4-1.dsc`), which uses another system by the late Joel Klecker (`<espy@debian.org>`).

Chapter 14

GnuPG

References:

- `gpg(1)`.
- `/usr/share/doc/gnupg/README.gz`
- *GNU privacy handbook* in `/usr/share/doc/gnupg-doc/GNU_Privacy_Handbook/` (install `gnupg-doc` package)

14.1 Installing GnuPG

```
# gpg --gen-key                      # generate a new key
# gpg --gen-revoke my_user_ID        # generate revoke key for my_user_ID
# host -l pgp.net | grep www|less # figure out pgp keyservers
```

As of now, good keyservers are:

```
keyserver wwwkeys.eu.pgp.net
keyserver wwwkeys.pgp.net
```

Here one must be careful **not to create more than 2 sub-keys**. If you do, keyservers on pgp.net will **corrupt** your key. Also, only one keyserver can be specified in `$HOME/.gnupg/options`.

Unfortunately, the following does not work any more:

```
keyserver search.keyserver.net
keyserver pgp.ai.mit.edu
```

14.2 Using GnuPG

File handling:

```
$ gpg  [options]  command  [args]
$ gpg {--armor|-a} {--sign|-s} file # sign file into a text file.asc
$ gpg --clearsign file              # clear-sign message
$ gpg --clearsign --not-dash-escaped patchfile  # clear-sign patchfile
$ gpg --verify file                 # verify clear-signed file
$ gpg -o file.sig {-b|--detach-sig} file # create detatched signature
$ gpg --verify file.sig file        # verify file with file.sig
$ gpg -o crypt_file {--recipient|-r} name {--encrypt|-e} file
        # public-key encryption intended for name
$ gpg -o crypt_file {--symmetric|-c} file # symmetric encryption
$ gpg -o file --decrypt crypt_file  # decryption
```

14.3 Managing GnuPG

Key management:

```
$ gpg --edit-key user_ID               # "help" for help, interactive
$ gpg -o file --exports                # export all keys to file
$ gpg --imports file                   # import all keys from file
$ gpg --send-keys user_ID              # send key of user_ID to keyserver
$ gpg --recv-keys user_ID              # recv. key of user_ID from keyserver
$ gpg --list-keys user_ID              # list keys of user_ID
$ gpg --list-sigs user_ID              # list sig. of user_ID
$ gpg --check-sigs user_ID             # check sig. of user_ID
$ gpg --fingerprint user_ID            # check fingerprint of user_ID
$ gpg --list-sigs | grep '^sig' | grep '[User id not found]' \
  | awk '{print $2}' | sort -u | xargs gpg --recv-keys # get unknown keys
  # update keys for all unknown sigs.
```

Trust code:

```
-         No ownertrust assigned / not yet calculated.
e         Trust calculation has failed.
q         Not enough information for calculation.
n         Never trust this key.
```

```
m         Marginally trusted.
f         Fully trusted.
u         Ultimately trusted.
```

The following will upload my key “*A8061F32*” to multiple key servers:

```
$ for xx in us es cz de dk uk ch net.uk earth.net.uk; \
$ do gpg --keyserver wwwkeys.$xx.pgp.net  --send-keys A8061F32; done
```

14.4 Using GnuPG with applications

14.4.1 Using GnuPG with Mutt

Add the following to `~/.muttrc` to keep a slow GnuPG from automatically starting, while allowing it to be used by typing ‘`S`’ at the index menu.

```
macro index S ":toggle pgp_verify_sig\n"
set pgp_verify_sig=no
```

14.4.2 Using GnuPG with Vim

Add the contents of `vimgpg` obtained from the examples subdirectory (`examples/`) into `~/.vimrc` to run GnuPG transparently.

Chapter 15

Support for Debian

The following resources provide help, advice, and support for Debian. Try your best to use self-help resources before crying out loud in the mailing lists. :)

Note that you can access a lot of documentation on your system by using a WWW browser, via the `dwww` or `dhelp` commands, found in their respective packages.

15.1 References

The following references are available for Debian and Linux in general. If their contents conflict with each other, always rely more on primary information sources than on secondary ones such as this document.

- Installation Manual (primary)
  - Read before installation and upgrade.
  - Web: `http://www.debian.org/releases/stable/installmanual`
  - Web: `http://www.debian.org/releases/testing/installmanual` (work in progress, sometimes this may not exist)
  - Package: `Not available in install-doc: Bug#155374`
  - File: `DebianCDunder/doc/`

- Release Notes (primary)
  - A must-read before installation and upgrade even if you are experienced.
  - Web: `http://www.debian.org/releases/stable/releasenotes`
  - Web: `http://www.debian.org/releases/testing/releasenotes` (work in progress)
  - Package: `Not available in install-doc: Bug#155374`
  - File: `DebianCDunder/doc/`

- FAQ (secondary)
  - Frequently asked questions
  - Web: `http://www.debian.org/doc/manuals/debian-faq/`
  - Package: `doc-debian`
  - File: `/usr/share/doc/debian/FAQ/index.html`

- Debian Reference (secondary)
  - Most comprehensive post-install user manual
  - Web: `http://www.debian.org/doc/manuals/debian-reference/`
  - Package: `debian-reference`
  - File: `/usr/share/doc/Debian/reference/`

- APT HOWTO (secondary)
  - Detailed user guide for Debian package management. (woody)
  - Web: `http://www.debian.org/doc/manuals/apt-howto/`
  - Package: `apt-howto`
  - File: `/usr/share/doc/Debian/apt-howto/`

- Securing Debian Manual (secondary)
  - Detailed user guide for securing and hardening of the default Debian installation. (woody)
  - Web: `http://www.debian.org/doc/manuals/securing-debian-howto/`
  - Package: `harden-doc`
  - File: `/usr/share/doc/harden-doc/html/securing-debian-howto/`

- `dselect` Documentation for Beginners (secondary)
  - Tutorial for `dselect`
  - Web: `http://www.debian.org/releases/woody/i386/dselect-beginner`
  - Package: `Not available in install-doc: Bug#155374`
  - File: `DebianCDunder/doc/`

- Debian Policy Manual (primary)
  - Technical backbone of Debian.
  - Web: `http://www.debian.org/doc/debian-policy/`
  - Package: `debian-policy`
  - File: `/usr/share/doc/debian-policy/`

- Debian Developer's Reference (primary)
  - Basic knowledge for developers.
  - The rest of us should also browse this once.
  - Web: `http://www.debian.org/doc/manuals/developers-reference/`
  - Package: `developers-reference`
  - File: `/usr/share/doc/developers-reference/`

- Debian New Maintainers' Guide (primary)

  - Practical guide for developers.
  - Packaging tutorials for the rest of us.
  - Web: http://www.debian.org/doc/manuals/maint-guide/
  - Package: `maint-guide`
  - File: `/usr/share/doc/maint-guide/`

- Packaging Manual (potato)
  - `packaging-manual` package in potato. (Moved into appendix of *Developer's Reference*)

- Unix-style manual pages (primary)
  - `man` *`package-name`*

- GNU-style info pages (primary)
  - `info` *`package-name`*

- Package specific documents (primary)
  - Find them under `/usr/share/doc/`*`package-name`*

- LDP: Linux Documentation Project (secondary)
  - General Linux HOWTOs and mini-HOWTOs
  - Web: http://www.tldp.org/
  - Package: `doc-linux`
  - File: `/usr/share/doc/HOWTO/`

- DDP: Debian Documentation Project (secondary)
  - Debian-specific manuals
  - Web: http://www.debian.org/doc/

- Debian Developers' Corner (secondary)
  - Key information for Debian developers
  - Insightful for end users
  - Web: http://www.debian.org/devel/

- Source code (absolutely primary)
  - No one can argue with this :-)
  - Download source code following 'The source code' on page 11

The following references are available for Unix in general. Please note that there are some minor differences between different Unix systems. Device names and init methods need extra attention.

- *The UNIX Programming Environment*
  - The book to read to learn about how UNIX works.
  - By B. W. Kernighan and R. Pike,

  - Published by Princeton Hall Software Series

- *The C Programming Language* (second edition)
  - The book to read to learn about ANSI C.
  - By B. W. Kernighan and D. M. Ritchie
  - Published by Princeton Hall Software Series

- *UNIX Power Tools*
  - The book to read to learn Unix tips.
  - By Jerry Peek, Tim O'Reilly and Mike Loukides
  - Published by O'Reilly and Associates

- *Essential System Administration* (second edition)
  - The book to read to learn about Unix system administration for many Unix flavors.
  - By Aeleen Frisch
  - Published by O'Reilly and Associates

- Bell Labs: Computing Sciences Research
  - Rich archive of Unix history
  - Main: `http://cm.bell-labs.com/cm/cs/`
  - Selected technical reports: `http://cm.bell-labs.com/cm/cs/cstr.html`
  - Some papers: `http://cm.bell-labs.com/cm/cs/papers.html`

- On-line Linux general support resources
  - Debian Planet (`http://www.debianplanet.org/`)
  - debianHELP (`http://www.debianhelp.org/`)
  - Linux.com (`http://linux.com/`)
  - The Linux Home Page at Linux Online (`http://www.linux.org/`)
  - Red Hat (commercial Linux vender) (`http://www.redhat.com/`) (RPM, Sys-V init)
  - SuSE, Inc. (commercial Linux vender) (`http://www.suse.de/`) (RPM, Sys-V init)
  - Slackware (`http://www.slackware.com/`) (TGZ, BSD-style init)

- On-line general Unix guide resources
  - A UNIX Introductory Course from Ohio State University (`http://www-wks.acs.ohio-state.edu/unix_course/unix.html`)
  - UNIXhelp from The University of Edinburgh (`http://www.ucs.ed.ac.uk/~unixhelp/servers.html`)
  - Unix / Programming Information (`http://arioch.unomaha.edu/~jclark/#info`)
  - comp.unix.questions FAQ (`http://www.faqs.org/faqs/unix-faq/faq/`)
  - comp.unix.user-friendly FAQ (`http://www.camelcity.com/~noel/usenet/cuuf-FAQ.htm`)
  - FreeBSD Documentation (`http://www.freebsd.org/docs.html`)
  - The FreeBSD Handbook (`http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/index.html`)

  - Matt Chapman's pages, UNIX GUIDE (`http://www.belgarath.demon.co.uk/`)

- Free software project home pages
  - GNU Project (`http://www.gnu.org/`)
  - The Linux Documentation Project (`http://www.tldp.org/`)
  - The Linux Kernel Archives (`http://www.linux.org/`)
  - The XFree86 Project, Inc (`http://www.xfree86.org/`)
  - GNOME (`http://www.gnome.org/`)
  - K Desktop Environment (`http://www.kde.org/`)
  - GNU software at Red Hat (`http://sources.redhat.com/`)
  - Mozilla (`http://www.mozilla.org`)
  - FreeBSD (`http://www.freebsd.org/`)
  - OpenBSD (`http://www.openbsd.org/`)
  - NetBSD (`http://www.netbsd.org/`)

15.2 Finding the meaning of a word

Many words used in Debian are cryptic jargon or acronyms. The following will solve most questions:

```
$ dict put-a-weird-word-here
```

15.3 Finding the popularity of a Debian package

Many packages exist in Debian and it is sometimes difficult to know which one to try first. See Debian Popularity Contest Results (`http://www.debian.org/~apenwarr/popcon/`) to get insight into what others are using. Also install the `popularity-contest` package to contribute.

15.4 The Debian bug tracking system

The Debian distribution has a bug tracking system (BTS) (`http://bugs.debian.org/`) which files details of bugs reported by users and developers. Each bug is given a number, and is kept on file until it is marked as having been dealt with.

You should check to see whether your bug report has already been filed by someone else before submitting it. Lists of currently outstanding bugs are available on the World Wide Web (`http://bugs.debian.org/`) and elsewhere (`http://www.debian.org/Bugs/Access`). See also 'Check bugs in Debian and seek help' on page 60.

The method of reporting a bug is described at `http://www.debian.org/Bugs/Reporting`.

15.5 Mailing lists

Read at least “debian-devel-announce” (English, read-only and low-traffic) to stay current with Debian.

The mailing lists of most interest to Debian users are “debian-user” (English, open and high-traffic) and other “debian-user-*language*” lists (for other languages).

For information on these lists and details of how to subscribe, see `http://lists.debian.org/`. Please check the archives for answers to your question prior to posting and also adhere to standard list etiquette.

15.6 Internet Relay Chat (IRC)

Debian has an IRC channel dedicated to the support and aid of Debian users located on the Open Projects IRC network, which is dedicated to providing collaborative information-sharing resources for the Open Source community. To access the channel point your favorite IRC client at irc.openprojects.net and join `#debian`.

Please follow the channel guidelines, respecting other users fully. For more information on Open Projects please visit the website (`http://www.openprojects.net`).

15.7 Search engines

There are many search engines that serve documentation related to Debian:

- Debian WWW search site (`http://search.debian.org/`).
- Google (`http://www.google.com/`): include “site:debian.org” as a search term.
- Google Groups (`http://groups.google.com/`): a search engine for newsgroups. Include “group:linux.debian.*” as a search term.
- AltaVista (`http://www.altavista.com/`)

For example, searching on the string “cgi-perl” gives a more detailed explanation of this package than the brief description field in its control file. See ‘Check bugs in Debian and seek help’ on page 60 for related advice.

15.8 Websites

The following are a few random URLs I collected for specific issues.

- Adrian Bunk's packages to run kernel 2.4.x on potato (`http://www.fs.tum.de/~bunk/kernel-24.html`)
- Linux on Laptops (`http://www.linux-laptop.net/`)
- Xterm FAQ (`http://dickey.his.com/xterm/xterm.faq.html`)
- EXT3 File System mini-HOWTO (`http://www.symonds.net/~rajesh/howto/ext3/index.html`)
- Large File Support in Linux (`http://www.suse.de/~aj/linux_lfs.html`)
- Window Managers for X (`http://www.xwinman.org`)
- Linux USB Project (`http://www.linux-usb.org/`)
- SuSE pages for CJK (`http://www.suse.de/~mfabian/suse-cjk/suse-cjk.html`)
- LNX-BBC (Business-card-sized boot CD project) (`http://www.lnx-bbc.org/`)
- Linux info by Karsten Self (partitioning, backup, browsers...) (`http://kmself.home.netcom.com/Linux/`)
- Backup info HOWTO by Alvin Oga (`http://www.Linux-Backup.net/`)
- Security info HOWTO by Alvin Oga (`http://www.Linux-Sec.net/`)
- Various UNOFFICIAL sources for APT (`http://www.internatif.org/bortzmeyer/debian/apt-sources/`)
- Laptop Ethernet Configuration (`http://www.orthogony.com/gjw/lap/lap-ether-intro.html`)

Appendix A

Appendix

A.1 Authors

Debian 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`>

QREF was short for the original document title, "Quick Reference. . . " and also is the project name at qref.sourceforge.net.

Most of the contents of 'Debian fundamentals' on page 5 originally came from "The Debian FAQ" (March 2002):

- 5. The Debian FTP archives: `ftparchives.sgml` (entire chapter)
- 6. Basics of the Debian Package Management System: `pkg_basics.sgml` (entire chapter)
- 7. The Debian Package Management Tools: `pkgtools.sgml` (entire chapter)
- 8. Keeping Your Debian System Up To Date: `uptodate.sgml` (entire chapter)
- 9. Debian and the kernel: `kernel.sgml` (entire chapter)
- 10. Customizing your installation of Debian GNU/Linux: `customizing.sgml` (part of chapter)

These sections of "The Debian FAQ" were included in this document after major reorganization to reflect changes in the Debian system. The content of this document is more recent.

The original "Debian FAQ" was made and maintained by J.H.M. Dassen (Ray) and Chuck Stickelman. Authors of the rewritten "Debian FAQ" are Susan G. Kleinmann and Sven Rudolph. After them, "The Debian FAQ" was maintained by Santiago Vila. The current maintainer is Josip Rodin.

Parts of the information for "The Debian FAQ" came from:

- The Debian-1.1 release announcement, by Bruce Perens (`http://www.perens.com/`).
- The Linux FAQ, by Ian Jackson (`http://www.chiark.greenend.org.uk/~ijackson/`).
- Debian Mailing List Archives (`http://lists.debian.org/`),
- the dpkg programmers' manual and the Debian Policy manual (see 'References' on page 165)
- many developers, volunteers, and beta testers, and
- the flaky memories of its authors. :-)

The authors would like to thank all those who helped make this document possible.

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.

Although I live in the USA, I am a non-native English user. Any grammatical corrections are welcomed.

The best feedback is a diff for the SGML version, but a diff for the text version is also welcomed. See 'Official document' on page 1 for the official document site.

The original SGML files used to create this document are also available in CVS at: `:pserver:anonymous@cvs.c` or `http://qref.sourceforge.net/Debian/qref.tar.gz`.

A.4 Document format

This document was written using the DebianDoc SGML DTD (rewritten from LinuxDoc SGML). The DebianDoc SGML system enables us to create files in a variety of formats from one source, e.g. this document can be viewed as HTML, plain text, TeX DVI, PostScript, PDF, or GNU info.

Conversion utilities for DebianDoc SGML are available in the Debian package `debiandoc-sgml`.

A.5 The Debian maze

The Linux system is a very powerful computing platform for a networked computer. However, learning how to use all its capabilities is not easy. Setting up the printer is a good example.

There is a complete, detailed map called the "SOURCE CODE". This is very accurate but very hard to understand. There are also references called HOWTO and mini-HOWTO. They are easier to understand but tend to give too much detail and lose the big picture. I sometimes have a problem finding the right section in a long HOWTO when I need a few commands to invoke.

In order to navigate through this maze of Linux system configuration, I started writing down simple reminder memos in text file format as my quick reference. This list of memos grew larger and I learned debiandoc in the meantime. The product is this *Debian Reference*.

A.6 The Debian quotes

Here are some interesting quotes from the Debian mailing list.

- “This is Unix. It gives you enough rope to hang yourself.” — Miquel van Smoorenburg `<miquels@cistron.nl>`
- “Unix **IS** user friendly… It’s just selective about who its friends are.” — Tollef Fog Heen `<tollef@add.no>`