The Debian GNU/Linux FAQ

February 14, 2026

The Debian GNU/Linux FAQ

Published October 2024

Contents

Abstract

This document answers questions frequently asked about Debian GNU/Linux.

Chapter 1

Definitions and overview

1.1 What is this FAQ?

This document gives frequently asked questions (with their answers!) about the Debian distribution (Debian GNU/Linux and others) and about the Debian project. If applicable, pointers to other documentation will be given: we won't quote large parts of external documentation in this document. You'll find out that some answers assume some knowledge of Unix-like operating systems. We'll try to assume as little prior knowledge as possible: answers to general beginners questions will be kept simple.

If you can't find what you're looking for in this FAQ, be sure to check out Section 12.1. If even that doesn't help, refer to Section 16.2.

1.2 What is Debian GNU/Linux?

Debian GNU/Linux is a particular *distribution* of the Linux operating system, and numerous packages that run on it.

Debian GNU/Linux is:

- **full featured**: Debian includes more than 64961 software packages at present. Users can select which packages to install; Debian provides a tool for this purpose. You can find a list and descriptions of the packages currently available in Debian at any of the Debian mirror sites (`https://www.debian.org/mirror/list`).
- **free to use and redistribute**: There is no consortium membership or payment required to participate in its distribution and development. All packages that are formally part of Debian GNU/Linux are free to redistribute, usually under terms specified by the GNU General Public License.

  The Debian archives also carry approximately 1082 software packages (in the `non-free` and `contrib` sections), which are distributable under specific terms included with each package.
- **dynamic**: With about 1400 volunteers constantly contributing new and improved code, Debian is evolving rapidly. The archives are updated twice every day.

Most Linux users run a specific *distribution* of Linux, like Debian GNU/Linux. However, in principle, users could obtain the Linux kernel via the Internet or from elsewhere, and compile it themselves. They could then obtain source code for many applications in the same way, compile the programs, then install them into their systems. For complicated programs, this process can be not only time-consuming but error-prone. To avoid it, users often choose to obtain the operating system and the application packages from one of the Linux distributors. What distinguishes the various Linux distributors are the software, protocols, and practices they use for packaging, installing, and tracking applications packages on users' systems, combined with installation and maintenance tools, documentation, and other services.

Debian GNU/Linux is the result of a volunteer effort to create a free, high-quality Unix-compatible operating system, complete with a suite of applications. The idea of a free Unix-like system originates from the GNU project, and many of the applications that make Debian GNU/Linux so useful were developed by the GNU project.

For Debian, free has the GNUish meaning (see the Debian Free Software Guidelines (`https://www.debian.org/social_contract#guidelines`)). When we speak of free software, we are referring

to freedom, not price. Free software means that you have the freedom to distribute copies of free software, that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

The Debian Project was created by Ian Murdock in 1993, initially under the sponsorship of the Free Software Foundation's GNU project. Today, Debian's developers think of it as a direct descendent of the GNU project.

Although Debian GNU/Linux itself is free software, it is a base upon which value-added Linux distributions can be built. By providing a reliable, full-featured base system, Debian provides Linux users with increased compatibility, and allows Linux distribution creators to eliminate duplication of effort and focus on the things that make their distribution special. See Section 14.3 for more information.

1.3 OK, now I know what Debian is... what is Linux?!

In short, Linux is the kernel of a Unix-like operating system. It was originally designed for 386 (and better) PCs; today Linux also runs on a dozen of other systems. Linux is written by Linus Torvalds and many computer scientists around the world.

Besides its kernel, a "Linux" system usually has:

- a file system that follows the Linux Filesystem Hierarchy Standard https://www.pathname.com/fhs.
- a wide range of Unix utilities, many of which have been developed by the GNU project and the Free Software Foundation.

The combination of the Linux kernel, the file system, the GNU and FSF utilities, and the other utilities are designed to achieve compliance with the POSIX (IEEE 1003.1) standard; see Section 4.4.

For more information about Linux, see What is Linux (https://www.linux.org/info/) by Linux Online (https://www.linux.org/).

1.4 Does Debian just do GNU/Linux?

Currently, Debian is only available for Linux, but with Debian GNU/Hurd we are offering a non-Linux-based OS as a development, server and desktop platform, too. However, this non-linux port is not officially released yet.

The Hurd is a set of servers running on top of the GNU Mach microkernel. Together they build the base for the GNU operating system.

Please see https://www.gnu.org/software/hurd for more information about the GNU/Hurd in general, and https://www.debian.org/ports/hurd/ for more information about Debian GNU/Hurd.

See https://www.debian.org/ports/#portlist-other for more information about hurd-amd64 and previous non-linux ports.

1.5 What is the difference between Debian GNU/Linux and other Linux distributions? Why should I choose Debian over some other distribution?

These key features distinguish Debian from other Linux distributions:

Freedom: As stated in the Debian Social Contract (https://www.debian.org/social_contract), Debian will remain 100% free. Debian is very strict about shipping truly free software. The guidelines used to determine if a work is "free" are provided in The Debian Free Software Guidelines (DFSG) (https://www.debian.org/social_contract#guidelines).

The Debian package maintenance system: The entire system, or any individual component of it, can be upgraded in place without reformatting, without losing custom configuration files, and (in most cases) without rebooting the system. Most Linux distributions available today have some kind of package maintenance system; the Debian package maintenance system is unique and particularly robust (see Chapter 7).

Open development: Whereas many other Linux distributions are developed by individuals, small, closed groups, or commercial vendors, Debian is a major Linux distribution that is being developed by an association of individuals who have made common cause to create a free operating system, in the same spirit as Linux and other free software.

More than 1000 volunteer package maintainers are working on over 64961 packages and improving Debian GNU/Linux. The Debian developers contribute to the project not by writing new applications (in most cases), but by packaging existing software according to the standards of the project, by communicating bug reports to upstream developers, and by providing user support. Additionally, more than 400 volunteer in the project doing other tasks. Examples of these tasks include: creating documentation, contributing to the website and translating. See also the additional information on how to become a contributor in Chapter 13.

The Universal Operating System: Debian comes with more than 64961 packages (`https://packages.debian.org/stable/`) and runs on 7 architectures (`https://www.debian.org/ports/`). This is far more than is available for any other GNU/Linux distribution. See Section 5.1 for an overview of the provided software and see Section 4.1 for a description of the supported hardware platforms.

The Bug Tracking System: The geographical dispersion of the Debian developers required sophisticated tools and quick communication of bugs and bug-fixes to accelerate the development of the system. Users are encouraged to send bugs in a formal style, which are quickly accessible by WWW archives or via e-mail. See additional information in this FAQ on the management of the bug log in Section 12.4.

The Debian Policy: Debian has an extensive specification of our standards of quality, the Debian Policy. This document defines the qualities and standards to which we hold Debian packages.

For additional information about this, please see our web page about reasons to choose Debian (`https://www.debian.org/intro/why_debian`).

1.6 How does the Debian project fit in or compare with the Free Software Foundation's GNU project?

The Debian system builds on the ideals of free software first championed by the Free Software Foundation (`https://www.gnu.org/`) and in particular by Richard Stallman (`https://www.stallman.org/`). FSF's powerful system development tools, utilities, and applications are also a key part of the Debian system.

The Debian Project is a separate entity from the FSF, however we communicate regularly and cooperate on various projects. The FSF explicitly requested that we call our system "Debian GNU/Linux", and we are happy to comply with that request.

1.7 How does one pronounce Debian and what does this word mean?

The project name is pronounced Deb'-ee-en, with a short e in Deb, and emphasis on the first syllable. This word is a contraction of the names of Debra and Ian Murdock, who founded the project. (Dictionaries seem to offer some ambiguity in the pronunciation of Ian (!), but Ian prefers ee'-en.)

Chapter 2

Getting and installing Debian GNU/Linux

The official document giving installation instructions is the Debian GNU/Linux Installation Guide (`https://www.debian.org/releases/stable/installmanual`). We'll give some additional notes about getting and installing Debian GNU/Linux here.

2.1 What is the latest version of Debian?

Currently there are three versions of Debian GNU/Linux:

release 13, a.k.a. the "stable" distribution or trixie This is stable and well tested software, it changes if major security or usability fixes are incorporated.

the "testing" distribution, currently called forky This is where packages that will be released as the next "stable" are placed; they've had some testing in unstable but they may not be completely fit for release yet. This distribution is updated more often than "stable", but not more often than "unstable".

the "unstable" distribution This is the version currently under development; it is updated continuously. You can retrieve packages from the "unstable" archive on any Debian mirror site and use them to upgrade your system at any time, but you may not expect the system to be as usable or as stable as before - that's why it's called "**unstable**"!

Please see Section 6.1 for more information.

2.2 Are there package upgrades in "stable"?

Generally speaking, no new functionality is added to the stable release. Once a Debian version is released and tagged "stable" most packages will only get security updates. That is, packages for which a security vulnerability has been found after the release will be upgraded. All the security updates are served through security.debian.org (`https://security.debian.org`).

However, there are some cases in which packages will be updated in stable. For example:

- When an urgent update is required to ensure the software continues working.
- The package is a data package and the data must be updated in a timely manner.
- The package needs to be current to useful to end user (e.g. some security software, such as anti-malware products).
- The software is a leaf package and is broken by changes external to the distribution.

Users that wish to run updated versions of the software in stable have the option to use "backports". Backports are recompiled packages from testing (mostly) and unstable (in a few cases only, e.g. security updates), so they will run without new libraries (wherever it is possible) on a stable Debian

distribution. Users can configure their system to use the backports repository and download specific software. However, it is recommended to pick out single backports which fit the specific needs, and not to use all backports available. For more information read the Wiki entry describing software available to Debian users (https://wiki.debian.org/DebianSoftware) and Wiki entry on backports (https://wiki.debian.org/Backports).

Security updates serve one purpose: to supply a fix for a security vulnerability. They are not a method for sneaking additional changes into the stable release without going through normal point release procedure. Consequently, fixes for packages with security issues will not upgrade the software. The Debian Security Team will backport the necessary fixes to the version of the software distributed in "stable" instead.

For more information related to security support please read the Security FAQ (https://www.debian.org/security/faq) or the Debian Security Manual (https://www.debian.org/doc/manuals/securing-debian-howto/).

2.3 Where/how can I get the Debian installation images?

Very likely, clicking the "Download" link on the main page of the Debian website (https://www.debian.org/) will get you what you need. However, for more options, get the various installation images from the Download Debian (https://www.debian.org/distrib/) page.

Please refer to Debian GNU/Linux on CDs (https://www.debian.org/CD) for more information about CD (and DVD) images.

2.4 How do I install Debian from CD-ROMs?

Installing Debian from CD is straightforward: configure your system for booting off a CD, insert your CD, and reboot. Your system will now be running the Debian Installer. See the Debian GNU/Linux Installation Guide (https://www.debian.org/releases/stable/installmanual) for more information.

2.5 Why does the official stable released CD-ROM contain symlinks for "frozen" and "unstable"? I thought this CD contains just "stable"!

Official Debian CD images indeed contain symlinks like:

```
/dists/frozen -> trixie/
/dists/stable -> trixie/
/dists/testing -> trixie/
/dists/unstable -> trixie/
```

so that they work when your sources.list has an entry like

```
deb cdrom:[<name as on cd label>]/ unstable main [...]
```

.

The fact these symlinks are present does *not* mean the image is "unstable" or "testing" or anything. Read the CD label in /.disk/info to find out which Debian version it contains. This information is also present in /README.txt on the CD.

Read https://www.debian.org/releases to find out what the current "stable" and "testing" releases are.

2.6 Can I get and install Debian directly from a remote Internet site?

Yes. You can boot the Debian installation system from a set of files you can download from our archive site and its mirrors.

You can download a small CD image file, create a bootable CD from it, install the basic system from it and the rest over the network. For more information please see https://www.debian.org/CD/netinst/.

2.7 Are there any alternative strategies for booting the system installer?

Yes. Apart from CD or DVD, you can install Debian GNU/Linux by booting from USB memory stick, directly from hard disk, or using TFTP net booting. For installing on multiple computers it's possible to do fully automatic installations. NB: not all methods are supported by all computer architectures. Once the installer has booted, the rest of the system can be downloaded over the network, or installed from local media. See the Debian GNU/Linux Installation Guide (https://www.debian.org/releases/stable/installmanual) for more information.

Chapter 3

Choosing a Debian distribution

There are many different Debian distributions. Choosing the proper Debian distribution is an important decision. This section covers some information useful for users that want to make the choice best suited for their system and also answers possible questions that might be arising during the process. It does not deal with ”why you should choose Debian” but rather ”which distribution of Debian”.

For more information on the available distributions read Section 6.1.

3.1 Which Debian distribution (stable/testing/unstable) is better for me?

The answer is a bit complicated. It really depends on what you intend to do. One solution would be to ask a friend who runs Debian. But that does not mean that you cannot make an independent decision. In fact, you should be able to decide once you complete reading this chapter.

- If security or stability are at all important for you: install stable. period. This is the most preferred way.
- If you are a new user installing to a desktop machine, start with stable. Some of the software is quite old, but it’s the least buggy environment to work in. You can easily switch to the more modern unstable (or testing) once you are a little more confident.
- If you are a desktop user with a lot of experience in the operating system and do not mind facing the odd bug now and then, or even full system breakage, use unstable. It has all the latest and greatest software, and bugs are usually fixed swiftly.
- If you are running a server, especially one that has strong stability requirements or is exposed to the Internet, install stable. This is by far the strongest and safest choice.

The following questions (hopefully) provide more detail on these choices. After reading this whole FAQ, if you still could not make a decision, stick with the stable distribution.

3.1.1 You asked me to install stable, but in stable so and so hardware is not detected/working. What should I do?

Try to search the web using a search engine and see if someone else is able to get it working in stable. Most of the hardware should work fine with stable. But if you have some state-of-the-art, cutting edge hardware, it might not work with stable. If this is the case, you might want to install/upgrade to either testing or unstable.

For laptops, https://www.linux-on-laptops.com/ is a very good website to see if someone else is able to get it to work under Linux. The website is not specific to Debian, but is nevertheless a tremendous resource. I am not aware of any such website for desktops.

Another option would be to ask in the debian-user mailing list by sending an email to debian-user@lists.debian.org. Messages can be posted to the list even without subscribing. The archives can be read through https://lists.debian.org/debian-user/. Information regarding subscribing to the list can be found at the location of archives. You are strongly encouraged to post your questions on

the mailing-list rather than on irc (`https://www.debian.org/support`). The mailing-list messages are archived, so the solution to your problem can help others with the same issue.

3.1.2 Will there be different versions of packages in different distributions?

Yes. Unstable has the most recent (latest) versions. But the packages in unstable are not well tested and might have bugs.

On the other hand, stable contains old versions of packages. But this package is well tested and is less likely to have any bugs.

The packages in testing fall between these two extremes.

3.1.3 The stable distributions really contains outdated packages. Just look at Kde, Gnome, Xorg or even the kernel. They are very old. Why is it so?

Well, you might be correct. The age of the packages at stable depends on when the last release was made. Since there is typically over 1 year between releases you might find that stable contains old versions of packages. However, they have been tested in and out. One can confidently say that the packages do not have any known severe bugs, security holes etc., in them. The packages in stable integrate seamlessly with other stable packages. These characteristics are very important for production servers which have to work 24 hours a day, 7 days a week.

On the other hand, packages in testing or unstable can have hidden bugs, security holes etc. Moreover, some packages in testing and unstable might not be working as intended. Usually people working on a single desktop prefer having the latest and most modern set of packages. Unstable is the solution for this group of people.

As you can see, stability and novelty are two opposing ends of the spectrum. If stability is required: install stable distribution. If you want to work with the latest packages, then install unstable.

3.1.4 If I were to decide to change to another distribution, can I do that?

Yes, but it is a one way process. You can go from stable --> testing --> unstable. But the reverse direction is not ”possible”. So better be sure if you are planning to install/upgrade to unstable.

Actually, if you are an expert and if you are willing to spend some time and if you are real careful and if you know what you are doing, then it might be possible to go from unstable to testing and then to stable. The installer scripts are not designed to do that. So in the process, your configuration files might be lost and...

3.1.5 Could you tell me whether to install stable, testing or unstable?

No. This is a rather subjective issue. There is no perfect answer as it depends on your software needs, your willingness to deal with possible breakage, and your experience in system administration. Here are some tips:

- Stable is rock solid. It does not break and has full security support. But it not might have support for the latest hardware.
- Testing has more up-to-date software than Stable, and it breaks less often than Unstable. But when it breaks, it might take a long time for things to get rectified. Sometimes this could be days and it could be months at times. It also does not have permanent security support.
- Unstable has the latest software and changes a lot. Consequently, it can break at any point. However, fixes get rectified in many occasions in a couple of days and it always has the latest releases of software packaged for Debian.

When deciding between testing and unstable bear in mind that there might be times when tracking testing would be beneficial as opposed to unstable. One of this document’s authors experienced such situation due to the gcc transition from gcc3 to gcc4. He was trying to install the `labplot` package on a machine tracking unstable and it could not be installed in unstable as some of its dependencies have undergone gcc4 transition and some have not. But the package in testing was installable on a testing machine as the gcc4 transitioned packages had not ”trickled down” to testing.

3.1.6 You are talking about testing being broken. What do you mean by that?

Sometimes, a package might not be installable through package management tools. Sometimes, a package might not be available at all, maybe it was (temporarily) removed due to bugs or unmet dependencies. Sometimes, a package installs but does not behave in the proper way.

When these things happen, the distribution is said to be broken (at least for this package).

3.1.7 Why is it that testing could be broken for months? Won't the fixes introduced in unstable flow directly down into testing?

The bug fixes and improvements introduced in the unstable distribution trickle down to testing after a certain number of days. Let's say this threshold is 5 days. The packages in unstable go into testing only when there are no RC-bugs reported against them. If there is a RC-bug filed against a package in unstable, it will not go into testing after the 5 days.

The idea is that, if the package has any problems, it would be discovered by people using unstable and will be fixed before it enters testing. This keeps testing in a usable state for most of the time. Overall a brilliant concept, if you ask me. But things aren't always that simple. Consider the following situation:

- Imagine you are interested in package XYZ.
- Let's assume that on June 10, the version in testing is XYZ-3.6 and in unstable it is XYZ-3.7.
- After 5 days, XYZ-3.7 from unstable migrates into testing.
- So on June 15, both testing and unstable have XYZ-3.7 in their repositories.
- Let's say, the user of testing distribution sees that a new XYZ package is available and updates the XYZ-3.6 to XYZ-3.7.
- Now on June 25, someone using testing or unstable discovers an RC bug in XYZ-3.7 and files it in the BTS.
- The maintainer of XYZ fixes this bug and uploads it to unstable say on June 30. Here it is assumed that it takes 5 days for the maintainer to fix the bug and upload the new version. The number 5 should not be taken literally. It could be less or more, depending upon the severity of the RC-bug at hand.
- This new version in unstable, XYZ-3.8 is scheduled to enter testing on July 5th.
- But on July 3rd some other person discovers another RC-bug in XYZ-3.8.
- Let's say the maintainer of XYZ fixes this new RC-bug and uploads new version of XYZ after 5 days.
- So on July 8th, testing has XYZ-3.7 while unstable has XYZ-3.9.
- This new version XYZ-3.9 is now rescheduled to enter testing on July 13th.
- Now since you are running testing, and since XYZ-3.7 is buggy, you could probably use XYZ only after July 13th. That is you essentially ended up with a broken XYZ for about one month.

The situation can get much more complicated, if say, XYZ depends on 4 other packages. This could in turn lead to an unusable testing distribution for months. While the scenario above is imaginary, similar things can occur in real life, though they are rare.

3.1.8 From an administrator's point of view, which distribution requires more attention?

One of the main reasons why many people choose Debian over other Linux distributions is that it requires very little administration. People want a system that just works. In general one can say that stable requires very little maintenance, while testing and unstable require constant maintenance from the administrator. If you are running stable, all you need to worry about is keeping track of security updates. If you are running either testing or unstable it is a good idea to be aware of the new bugs discovered in the installed packages, new bugfixes/features introduced etc.

3.1.9 What happens when a new release is made?

This question will not help you in choosing a Debian distribution. But sooner or later you will face this question.

The stable distribution is currently trixie; The next stable distribution will be called forky. Let's consider the particular case of what happens when forky is released as the new stable version.

- oldstable = bookworm; stable = trixie; testing = forky; unstable = sid
- Unstable is always referred to as sid irrespective of whether a release is made or not.
- Packages constantly migrate from sid to testing (i.e. forky). But packages in stable (i.e. trixie) remain the same except for security updates.
- After some time testing becomes frozen. But it will still be called testing. At this point no new packages from unstable can migrate to testing unless they include release-critical (RC) bug fixes.
- When testing is frozen, all the new bugfixes introduced have to be manually checked by the members of the release team. This is done to ensure that there won't be any unknown severe problems in the frozen testing.
- RC bugs in 'frozen testing' are reduced to either zero or, if greater than zero, the bugs are either marked as ignored for the release or are deferred for a point release.
- The 'frozen testing' with no rc-bugs will be released as the new stable version. In our example, this new stable release will be called forky.
- At this stage oldstable = trixie, stable = forky. The contents of stable and 'frozen testing' are same at this point.
- A new testing is based on the old testing.
- Packages start coming down from sid to testing and the Debian community will be working towards making the next stable release.

3.1.10 I have a working Desktop/cluster with Debian installed. How do I know which distribution I am running?

In most situations it is very easy to figure this out. Take a look at the `/etc/apt/sources.list` file. There will be an entry similar to this:

```
deb http://deb.debian.org/debian/ unstable main contrib
```

The third field ('unstable' in the above example) indicates the Debian distribution the system is currently tracking.

You can also use **lsb_release** (available in the `lsb-release` package). If you run this program in an unstable system you will get:

```
$ lsb_release  -a
LSB Version:    core-2.0-noarch:core-3.0-noarch:core-3.1-noarch:core-2.0-ia32: ↩
    core-3.0-ia32:core-3.1-ia32
Distributor ID: Debian
Description:    Debian GNU/Linux unstable (sid)
Release:    unstable
Codename:   sid
```

However, this is not always that easy. Some systems might have `sources.list` files with multiple entries corresponding to different distributions. This could happen if the administrator is tracking different packages from different Debian distributions. This is frequently referred to as apt-pinning. These systems might run a mixture of distributions.

3.1.11 I am currently using the stable version. Can I change to testing or unstable? If so, how?

First of all, please bear in mind that the stable version is the one recommended for servers as well as for desktop computers, not only you will get security updates if you are running stable but also there are less changes which could potentially break the system or your setup.

If you are currently running stable, then in the `/etc/apt/sources.list` file the third field will be either 'trixie' or 'stable'. If you want to change to a different version, you need to change this to the distribution you want to run. If you want to run testing, then change the third field of `/etc/apt/sources.list` to 'testing'. If you want to run unstable, then change the third field to 'unstable'.

Currently testing is called forky. So, if you change the third field of `/etc/apt/sources.list` to 'forky', then also you will be running testing. But even when forky becomes stable, you will still be tracking forky.

Unstable is always called Sid. So if you change the third field of `/etc/apt/sources.list` to 'sid', then you will be tracking unstable.

Currently Debian offers does not offer security updates for testing nor for unstable. The Debian Security Team (`https://security-team.debian.org/`) focus their work on stable and old-stable. Nevertheless, just like any other type of fixes, security fixes in unstable are directly made to the main archive and trickle down to testing in due course. So if you are running unstable make sure that you remove the lines relating to security updates in `/etc/apt/sources.list`. If you are interested in knowing what known security bugs exist in these versions of the distributions, you will this information in the list of vulnerable source packages in testing (`https://security-tracker.debian.org/tracker/status/release/testing`), and unstable (`https://security-tracker.debian.org/tracker/status/release/unstable`).

If there is a release notes document available for the distribution you are upgrading to (even though it has not yet been released) it would be wise to review it, as it might provide information on how you should upgrade to it. You will always find the Release Notes for testing (`https://www.debian.org/releases/testing/releasenotes`) available at the Debian website but depending on how close the testing version is to release the document might not cover all the potential changes or pitfalls.

Nevertheless, once you make the above changes, you can run `aptitude update` and then install the packages that you want. Notice that installing a package from a different distribution might automatically upgrade half of your system. If you install individual packages you will end up with a system running mixed distributions.

It might be best in some situations to just fully upgrade to the new distribution running **apt full-upgrade**, **aptitude safe-upgrade** or **aptitude full-upgrade**. Read apt's and aptitude's manual pages for more information.

The Debian reference manual provides more insight on running testing and unstable in its section Life with eternal upgrades (`https://www.debian.org/doc/manuals/debian-reference/ch02.en.html#_life_with_eternal_upgrades`).

3.1.12 I am currently tracking testing (forky). What will happen when a release is made? Will I still be tracking testing or will my machine be running the new stable distribution?

It depends on the entries in the `/etc/apt/sources.list` file. If you are currently tracking testing, these entries are similar to either:

```
deb http://deb.debian.org/debian/ testing main
```

or

```
deb http://deb.debian.org/debian/ forky main
```

If the third field in `/etc/apt/sources.list` is 'testing' then you will be tracking testing even after a release is made. So after forky is released, you will be running a new Debian distribution which will have a different codename. Changes might not be apparent at first but will be evident as soon as new packages from unstable go over to the testing distribution.

But if the third field contains 'forky' then you will be tracking stable (since forky will then be the new stable distribution).

3.1.13 I am still confused. What did you say I should install?

If unsure, the best bet would be the stable distribution.

3.2 But what about Kali, Knoppix, Linux Mint, Ubuntu, and others?

These distributions are not Debian; they are *Debian-based*. Though there are many similarities and commonalities between them, there are also crucial differences.

Over the years, many distributions have been developed over time reusing and/or rebuilding Debian packages and also adding custom packages on their own. Most of the distributions have been created for specific audiences or purposes. According to Distrowatch, Debian has spawned more than 420 derivatives (`https://distrowatch.com/search.php?basedon=Debian&status=All#distrosearch`) and more than 120 of them are active at the time of writing.

All these distributions have their own merits and are suited to some specific set of users. For more information, read Debian derivatives (`https://www.debian.org/misc/children-distros`) available at the Debian website. You can find a complete list of Debian derivatives, including those that are no longer under active development at the Debian derivate Census (`https://wiki.debian.org/Derivatives/Census`) in the Wiki.

3.2.1 I know that Kali/Knoppix/Linux Mint/Ubuntu/... is Debian-based. So after installing it on the hard disk, can I use 'apt' package tools on it?

These distributions are Debian-based, but they are not Debian. You will be still able to use apt package tools by pointing the `/etc/apt/sources.list` file to these distributions' repositories. In some cases some distributions might even have additional package managers that are used instead of apt.

In most situations if you stick with one distribution you should use that and not mix packages from other distributions. Many common breakages arise due to people running a distribution and trying to install Debian packages from other distributions. The fact that they use the same formatting and name (.deb), does not make them immediately compatible.

For example, Knoppix is a Linux distribution designed to be booted as a live CD whereas Debian is designed to be installed on the hard-disk. Knoppix is great if you want to know whether a particular piece of hardware works, or if you want to experience how a GNU/Linux system 'feels' etc., Knoppix is good for demonstration purposes while Debian is designed to run 24/7. Moreover the number of packages available and the number of architectures supported by Debian are far more than that of Knoppix.

If you want Debian, it is best to install Debian from the get-go. Although it is possible to install Debian through other distributions, such as Knoppix, the procedure calls for expertise. If you are reading this FAQ, I would assume that you are new to both Debian and Knoppix. In that case, save yourself a lot of trouble later and install Debian right from the beginning.

3.2.2 I installed Kali/Knoppix/Linux Mint/Ubuntu/... on my hard disk. Now I have a problem. What should I do?

You are advised not to use the Debian forums (either mailing lists or IRC) for help on Debian derivatives as people there may base their suggestions on the assumption that you are running a Debian system. These "fixes" might not be suited to what you are running, and might even make your problem worse.

Use the forums of the specific distribution you are using first. If you do not get help or the help you get does not fix your problem you might want to try asking in Debian forums, but keep the advice of the previous paragraph in mind.

3.2.3 I'm using Kali/Knoppix/Linux Mint/Ubuntu/... and now I want to use Debian. How do I migrate?

Consider the change from a Debian-based distribution to Debian just like a change from one operating system to another one. You should make a backup of all your data and reinstall the operating system

from scratch. You should not attempt to ”upgrade” to Debian using the package management tools as you might end up with an unusable system.

If all your user data (i.e. your `/home`) is under a separate partition migrating to Debian is actually quite simple, you just have to tell the installation system to mount (but not reformat) that partition when reinstalling. Making backups of your data, as well as your previous system’s configuration (i.e. `/etc/` and, maybe, `/var/`) is still encouraged.

Chapter 4

Compatibility issues

4.1 On what hardware architectures/systems does Debian GNU/Linux run?

Debian GNU/Linux includes complete source-code for all of the included programs, so it should work on all systems which are supported by the Linux kernel; see the Linux FAQ (`http://en.tldp.org/FAQ/Linux-FAQ/intro.html#DOES-LINUX-RUN-ON-MY-COMPUTER`) for details.

The current Debian GNU/Linux release, 13, contains a complete, binary distribution for the following architectures:

- *amd64*: this covers systems based on AMD 64bit CPUs with AMD64 extension and all Intel CPUs with EM64T extension, and a common 64bit userspace.
- *arm64*: supports the latest 64-bit ARM-powered devices.
- *armel*: little-endian ARM machines.
- *armhf*: an alternative to armel for ARMv7 machines with hard-float.
- *i386*: this covers systems based on Intel and compatible processors, including Intel's 386, 486, Pentium, Pentium Pro, Pentium II (both Klamath and Celeron), and Pentium III, and most compatible processors by AMD, Cyrix and others.
- *ia64*: Intel IA-64 ("Itanium") computers.
- *mips*: SGI's big-endian MIPS systems, Indy and Indigo2; *mipsel*: little-endian MIPS machines, Digital DECstations.
- *powerpc*: this covers some IBM/Motorola PowerPC machines, including the Apple Macintosh PowerMac models, and the CHRP and PReP open architecture machines.
- *ppc64el*: 64-bit little-endian PowerPC port, supports several recent PowerPC/POWER processors.
- *s390x*: 64-bit port for IBM System z machines, replaced s390.

The development of binary distributions of Debian for *hurd-i386* (for GNU Hurd kernel on i386 32-bit PCs), *mipsel64* (for 64 bit MIPS in little-endian mode), *powerpcspe* (port for the "Signal Processing Engine" hardware), *sparc64* (for 64 bit SPARC processors), *sh* (for Hitachi SuperH processors), and *x32* (for amd64/x86_64 CPUs using 32-bit pointers) is currently underway.

Support for the *m68k* architecture was dropped in the Etch (Debian 4.0) release, because it did not meet the criteria set by the Debian Release Managers. This architecture covers Amigas and ATARIs having a Motorola 680x0 processor for $x >= 2$; with MMU. However, the port is still active and available for installation even if not a part of this official stable release and might be reactivated for future releases.

Support for the *hppa* (Hewlett-Packard's PA-RISC machines) and *alpha* (Compaq/Digital's Alpha systems) were dropped in the Squeeze (Debian 6.0) release for similar reasons. The *arm* was dropped too in this release, as it was superseded by the *armel* architecture.

Support for the 32-bit *s390* port (s390) was discontinued and replaced with s390x in Jessie (Debian 8). In addition, the ports to IA-64 and Sparc had to be removed from this release due to insufficient developer support.

For more information on the available ports see the ports pages at the website (`https://www.debian.org/ports/`).

For further information on booting, partitioning your drive, enabling PCMCIA (PC Card) devices and similar issues please follow the instructions given in the Installation Manual, which is available from our WWW site at `https://www.debian.org/releases/stable/installmanual`.

4.2 What kernels does Debian GNU/Linux run?

Beside Linux, Debian provides a complete, binary distribution for the following operating system kernels:

- FreeBSD: provided through the *kfreebsd-amd64* and *kfreebsd-i386* ports, for 64-bit PCs and 32-bit PCs respectively. These ports were first released in Debian 6.0 Squeeze as a *technology preview*. However they were not part of the Debian 8 Jessie release.

In addition to these, work is in progress on the following adaptations:

- *avr32*, port to Atmel's 32-bit RISC architecture,
- *hurd-i386*, a port for 32-bit PC. This port will use GNU Hurd, the new operating system being put together by the GNU group,
- *sh*, port to Hitachi SuperH processors.

There were attempts to port the distribution to the NetBSD kernel, providing *netbsd-i386* (for 32-bit PCs) and *netbsd-alpha* (for Alpha machines) but these ports were never released and are currently abandoned.

For more information on the available ports see the ports pages at the website (`https://www.debian.org/ports/`).

4.3 How compatible is Debian with other distributions of Linux?

Debian developers communicate with other Linux distribution creators in an effort to maintain binary compatibility across Linux distributions.[1] Most commercial Linux products run as well under Debian as they do on the system upon which they were built.

Debian GNU/Linux adheres to the Linux Filesystem Hierarchy Standard (`https://www.pathname.com/fhs`). However, there is room for interpretation in some of the rules within this standard, so there may be slight differences between a Debian system and other Linux systems.

4.4 How source code compatible is Debian with other Unix systems?

For most applications Linux source code is compatible with other Unix systems. It supports almost everything that is available in System V Unix systems and the free and commercial BSD-derived systems. However in the Unix business such claim has nearly no value because there is no way to prove it. In the software development area complete compatibility is required instead of compatibility in "about most" cases. So years ago the need for standards arose, and nowadays POSIX.1 (IEEE Standard 1003.1-1990) is one of the major standards for source code compatibility in Unix-like operating systems.

Linux is intended to adhere to POSIX.1, but the POSIX standards cost real money and the POSIX.1 (and FIPS 151-2) certification is quite expensive; this made it more difficult for the Linux developers to work on complete POSIX conformance. The certification costs make it unlikely that Debian will get an

[1] The Linux Standard Base (`https://wiki.linuxfoundation.org/lsb/start/`) is a specification for allowing the same binary package to be used on multiple distributions. After Jessie (Debian 8) was released, Debian abandoned (`https://sources.debian.org/src/lsb/9.20170808/debian/README.Debian/`) the pursuit of LSB compatibility. See this July 3, 2015 message from Didier Raboud (`https://lists.debian.org/4526217.myWFlvm1rM@gyllingar`) and the following discussion for background information.

official conformance certification even if it completely passed the validation suite. (The validation suite is now freely available, so it is expected that more people will work on POSIX.1 issues.)

Unifix GmbH (Braunschweig, Germany) developed a Linux system that has been certified to conform to FIPS 151-2 (a superset of POSIX.1). This technology was available in Unifix' own distribution called Unifix Linux 2.0 and in Lasermoon's Linux-FT.

4.5 Can I use Debian packages (".deb" files) on my Red Hat/Slackware/... Linux system? Can I use Red Hat packages (".rpm" files) on my Debian GNU/Linux system?

Different Linux distributions use different package formats and different package management programs.

You probably can: A program to unpack a Debian package onto a Linux host that is been built from a "foreign" distribution is available, and will generally work, in the sense that files will be unpacked. The converse is probably also true, that is, a program to unpack a Red Hat or Slackware package on a host that is based on Debian GNU/Linux will probably succeed in unpacking the package and placing most files in their intended directories. This is largely a consequence of the existence (and broad adherence to) the Linux Filesystem Hierarchy Standard. The Alien (`https://packages.debian.org/alien`) package is used to convert between different package formats.

You probably do not want to: Most package managers write administrative files when they are used to unpack an archive. These administrative files are generally not standardized. Therefore, the effect of unpacking a Debian package on a "foreign" host will have unpredictable (certainly not useful) effects on the package manager on that system. Likewise, utilities from other distributions might succeed in unpacking their archives on Debian systems, but will probably cause the Debian package management system to fail when the time comes to upgrade or remove some packages, or even simply to report exactly what packages are present on a system.

A better way: The Linux File System Standard (and therefore Debian GNU/Linux) requires that subdirectories under `/usr/local/` be entirely under the user's discretion. Therefore, users can unpack "foreign" packages into this directory, and then manage their configuration, upgrade and removal individually.

4.6 How should I install a non-Debian program?

Files under the directory `/usr/local/` are not under the control of the Debian package management system. Therefore, it is good practice to place the source code for your program in /usr/local/src/. For example, you might extract the files for a package named "foo.tar" into the directory `/usr/local/src/foo`. After you compile them, place the binaries in `/usr/local/bin/`, the libraries in `/usr/local/lib/`, and the configuration files in `/usr/local/etc/`.

If your programs and/or files really must be placed in some other directory, you could still store them in `/usr/local/`, and build the appropriate symbolic links from the required location to its location in `/usr/local/`, e.g., you could make the link

```
ln -s /usr/local/bin/foo /usr/bin/foo
```

In any case, if you obtain a package whose copyright allows redistribution, you should consider making a Debian package of it, and uploading it for the Debian system. Guidelines for becoming a package developer are included in the Debian Policy manual (see Section 12.1).

Chapter 5

Software available in the Debian system

5.1 What types of applications and development software are available for Debian GNU/Linux?

Like most Linux distributions, Debian GNU/Linux provides:

- the major GNU applications for software development, file manipulation, and text processing, including gcc, g++, make, texinfo, Emacs, the Bash shell and numerous upgraded Unix utilities,
- Perl, Python, Tcl/Tk and various related programs, modules and libraries for each of them,
- TeX (LaTeX) and Lyx, dvips, Ghostscript,
- the Xorg windowing system, which provides a networked graphical user interface for Linux, and countless X applications including the GNOME, KDE and Xfce desktop environments,
- a full suite of networking applications, including servers for Internet protocols such as HTTP (WWW), FTP, NNTP (news), SMTP and POP (mail) and DNS (name servers); relational databases like PostgreSQL, MySQL; also provided are web browsers including the various Mozilla products,
- a complete set of office applications, including the LibreOffice productivity suite, Gnumeric and other spreadsheets, WYSIWYG editors, calendars.

More than 63879 packages, ranging from news servers and readers to sound support, FAX programs, database and spreadsheet programs, image processing programs, communications, net, and mail utilities, Web servers, and even ham-radio programs are included in the distribution. Other 1082 software suites are available as Debian packages, but are not formally part of Debian due to license restrictions.

5.2 Who wrote all that software?

For each package the *authors* of the program(s) are credited in the file `/usr/share/doc/PACKAGE/copyright`, where PACKAGE is to be substituted with the package's name.

Maintainers who package this software for the Debian GNU/Linux system are listed in the Debian control file (see Section 7.4) that comes with each package. The Debian changelog, in `/usr/share/doc/PACKAGE/cha` mentions the people who've worked on the Debian packaging too.

5.3 How can I get a current list of programs that have been packaged for Debian?

A complete list is available from any of the Debian mirrors (`https://www.debian.org/mirror/list`), in the file `indices/Maintainers`. That file includes the package names and the names and e-mails of their respective maintainers.

The WWW interface to the Debian packages (`https://packages.debian.org/`) conveniently summarizes the packages in each of about thirty "sections" of the Debian archive.

5.4 How can I install a developer's environment to build packages?

If you want to build packages in your Debian system you will need to have a basic development environment, including a C/C++ compiler and some other essential packages. In order to install this environment you just need to install the `build-essential` package. This is a meta-package or placeholder package which depends on the standard development tools one needs to build a Debian package.

Some software can, however, need additional software to be rebuilt, including library headers or additional tools such as **autoconf** or **gettext**. Debian provides many of the tools needed to build other software as Debian packages.

Finding which software is precisely required can be tricky, however, unless you are planning on rebuilding Debian packages. This last task is rather easy to do, as official packages have to include a list of the additional software (besides the packages in `build-essential`) needed to build the package, this is known as `Build-Dependencies`. To install all the packages needed to build a given source package and then build said source package you can just run:

```
# apt-get build-dep foo
# apt-get source --build foo
```

Notice that if you want to build the Linux kernels distributed by Debian you will want to install also the `kernel-package` package. For more information see Section 10.2.

5.5 What is missing from Debian GNU/Linux?

There is a list of packages which still need to be packaged for Debian, the Work-Needing and Prospective Packages list (`https://www.debian.org/devel/wnpp/`).

For more details about adding missing things, see Chapter 13.

5.6 Why do I get "ld: cannot find -lfoo" messages when compiling programs? Why aren't there any libfoo.so files in Debian library packages?

Debian Policy requires that such symbolic links (to libfoo.so.x.y.z or similar) are placed in separate, development packages. Those packages are usually named libfoo-dev or libfooX-dev (presuming the library package is named libfooX, and X is a whole number).

5.7 (How) Does Debian support Java?

Several *free* implementations of Java technology are available as Debian packages, providing both Java Development Kits as well as Runtime Environments. You can write, debug and run Java programs using Debian.

Running a Java applet requires a web browser with the capability to recognize and execute it. Several web browsers available in Debian, such as Mozilla or Konqueror, support Java plug-ins that enable running Java applets within them.

Please refer to the Debian Java FAQ (`https://www.debian.org/doc/manuals/debian-java-faq/`) for more information.

5.8 How can I check that I am using a Debian system, and what version it is?

In order to make sure that your system has been installed from the real Debian base disks, use the

```
lsb_release -a
```

command. It will display the name of the distribution (in Distributor ID field) and the version of the system (in Release and Codename fields). The following is an example run in a Debian system:

```
$ lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description:    Debian GNU/Linux 7.4 (wheezy)
Release:    7.4
Codename:   wheezy
```

You can also check for the existence of `/etc/debian_version` file, which contains a single one-line entry giving the version number of the release, as defined by the package `base-files`.

Users should be aware, however, that the Debian system consists of many parts, each of which can be updated (almost) independently. Each Debian "release" contains well defined and unchanging contents. Updates are separately available. For a one-line description of the installation status of package `foo`, use the command `dpkg --list foo`. For a more verbose description, use:

```
dpkg --status foo
```

To view versions of all installed packages, run:

```
dpkg -l
```

Note that the existence of the program `dpkg` shows that you should be able to install Debian packages on your system. However, since the program has been ported to many other operating systems and architectures, this is no longer a reliable method of determining if a system is Debian GNU/Linux.

5.9 How does Debian support non-English languages?

- Debian GNU/Linux is distributed with keymaps for nearly two dozen keyboards, and with utilities (in the `kbd` package) to install, view, and modify the tables.

  The installation prompts the user to specify the keyboard to use.

- Nearly all of the software in Debian supports UTF-8 as character set. Legacy character sets, such as ISO-8859-1 or ISO-8859-2, should be considered obsolete.

- Currently, support for German-, Spanish-, French-, Hungarian-, Italian-, Japanese-, Korean-, Dutch-, Polish-, Portuguese-, Russian-, Turkish-, and Chinese-language manual pages 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 'italian'. The **man** program will then search for Italian manual pages under `/usr/share/man/it/`.

5.10 I have a device that requires non-free firmware. What should I do?

Firmware refers to embedded software which controls electronic devices. Many devices require firmware to operate. Historically, firmware would be built into the device's ROM or Flash memory, but more and more often, a firmware image has to be loaded into the device RAM by a device driver during device initialisation.

Some devices require non-free firmware to work properly. And there could be firmware updates in the future.

In Debian release 11 (bullseye) and older, Debian did not include non-free firmware on official images and live installations. For Debian 12 onwards, all the packaged non-free firmware binaries that Debian can distribute have been moved to a new component in the Debian archive, called `non-free-firmware`. If you're upgrading from an older release of Debian and you need these firmware binaries, make sure that your sources.list uses this new component:

```
http://deb.debian.org/debian trixie main non-free-firmware contrib non-free
```

For more information please refer to Firmware information (`https://wiki.debian.org/Firmware`) in the Debian Wiki.

5.11 I have a wireless network card which doesn't work with Linux. What should I do?

Buy one which does :)

Alternatively, use `ndiswrapper` to use a driver for Windows (if you have one) on your Linux system. See the Debian Wiki ndiswrapper page (`https://wiki.debian.org/NdisWrapper`) for more information.

Chapter 6

The Debian archives

6.1 How many Debian distributions are there?

There are three major distributions: the ”stable” distribution, the ”testing” distribution, and the ”unstable” distribution. The ”testing” distribution is sometimes ”frozen” (see Section 6.5.1). Next to these, there is the ”oldstable” distribution (that’s just the one from before ”stable”), and the ”experimental” distribution.

Experimental is used for packages which are still being developed, and with a high risk of breaking your system. It’s used by developers who’d like to study and test bleeding edge software. Users shouldn’t be using packages from there, because they can be dangerous and harmful even for the most experienced people.

See Chapter 3 for help when choosing a Debian distribution.

6.2 What are all those names like etch, lenny, etc.?

They are just ”codenames”. When a Debian distribution is in the development stage, it has no version number but a codename. The purpose of these codenames is to make easier the mirroring of the Debian distributions (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 `trixie` (i.e. Debian GNU/Linux 13) and `testing` is a symbolic link to `forky`. This means that `trixie` is the current stable distribution and `forky` is the current testing distribution.

`unstable` is a permanent symbolic link to `sid`, as `sid` is always the unstable distribution (see Section 6.3).

6.2.1 Which other codenames have been used in the past?

Aside `trixie` and `forky`, other codenames that have been already 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, `potato` for release 2.2, `woody` for release 3.0, `sarge` for release 3.1, `etch` for release 4.0, `lenny` for release 5.0, `squeeze` for release 6.0, `wheezy` for release 7, `jessie` for release 8, `stretch` for release 9, `buster` for release 10, `bullseye` for release 11, `bookworm` for release 12.

6.2.2 Where do these codenames come from?

So far they have been characters taken from the ”Toy Story” movies by Pixar.

- *buzz* (Debian 1.1) was the spaceman Buzz Lightyear,
- *rex* (Debian 1.2) was the tyrannosaurus,
- *bo* (Debian 1.3) was Bo Peep, the girl who took care of the sheep,
- *hamm* (Debian 2.0) was the piggy bank,

- *slink* (Debian 2.1) was Slinky Dog, the toy dog,
- *potato* (Debian 2.2) was, of course, Mr. Potato,
- *woody* (Debian 3.0) was the cowboy,
- *sarge* (Debian 3.1) was the sergeant of the Green Plastic Army Men,
- *etch* (Debian 4.0) was the toy whiteboard (Etch-a-Sketch),
- *lenny* (Debian 5.0) was the toy binoculars,
- *squeeze* (Debian 6) was the name of the three-eyed aliens,
- *wheezy* (Debian 7) was the rubber toy penguin with a red bow tie,
- *jessie* (Debian 8) was the yodeling cowgirl,
- *stretch* (Debian 9) was the rubber toy octopus with suckers on her eight long arms.
- *buster* (Debian 10) was Andy’s pet dog.
- *bullseye* (Debian 11) was Woody’s wooden toyhorse.
- *bookworm* (Debian 12) was a green toy worm with a built-in flashlight who loves reading books.
- *trixie* (Debian 13) was a blue plastic triceratops.
- *sid* was the evil neighbor kid next door who broke all toys.

The decision (`https://lists.debian.org/debian-devel/1996/06/msg00515.html`) of using Toy Story names was made (`https://lists.debian.org/debian-user/1997/04/msg00011.html`) by Bruce Perens who was, at the time, the Debian Project Leader and was working also at Pixar, the company that produced the movies.

6.3 What about ”sid”?

sid or *unstable* is the place where most of the packages are initially uploaded. It will never be released directly, because packages which are to be released will first have to be included in *testing*, in order to be released in *stable* later on. sid contains packages for both released and unreleased architectures.

The name ”sid” also comes from the ”Toy Story” animated motion picture: Sid was the boy next door who destroyed toys :-)
[1]

6.4 What does the stable directory contain?

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

  These packages all comply with the Debian Free Software Guidelines (`https://www.debian.org/social_contract#guidelines`), and are all freely usable and distributable.

[1] When the present-day sid did not exist, the FTP site organization had one major flaw: there was an assumption that when an architecture is created in the current unstable, it will be released when that distribution becomes the new stable. For many architectures that isn’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 normal. This layout was somewhat confusing to users.

With the advent of package pools (see Section 6.10), 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).

- 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.

6.5 What does the testing distribution contain?

Packages are installed into the ”testing” directory after they have undergone some degree of testing in unstable.

They must be in sync on all architectures where they have been built and mustn’t have dependencies that make them uninstallable; they also need 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 information about the status of ”testing” in general and the individual packages is available at `https://www.debian.org/devel/testing`.

6.5.1 What about ”testing”? How is it ”frozen”?

When the ”testing” distribution is mature enough, the release manager starts ”freezing” it. The normal propagation delays are increased to ensure that as few new bugs as possible from ”unstable” enter ”testing”.

After a while, the ”testing” distribution becomes truly ”frozen”. This means that all new packages that are to propagate to the ”testing” are held back, unless they include release-critical bug fixes. The ”testing” distribution can also remain in such a deep freeze during the so-called ”test cycles”, when the release is imminent.

When a ”testing” release becomes ”frozen”, ”unstable” tends to partially freeze as well. This is because developers are reluctant to upload radically new software to unstable, in case the frozen software in testing needs minor updates and to fix release critical bugs which keep testing from becoming ”stable”.

We keep a record of bugs in the ”testing” distribution that can hold off a package from being released, or bugs that can hold back the whole release. For details, please see current testing release information (`https://www.debian.org/releases/testing/`).

Once that bug count lowers to maximum acceptable values, the frozen ”testing” distribution is declared ”stable” and released with a version number.

The most important bug count is the ”Release Critical” bug count, which can be followed in the Release-critical bug status page (`https://bugs.debian.org/release-critical/`). A common release goal is NoRCBugs (`https://wiki.debian.org/ReleaseGoals/NoRCBugs`) which means that the distribution should not have any bugs of severity critical, grave or serious. The full list of issues considered critical can be found in the RC policy document (`https://release.debian.org/testing/rc_policy.txt`).

With each new release, the previous ”stable” distribution becomes obsolete and moves to the archive. For more information please see Debian archive (`https://www.debian.org/distrib/archive`).

6.6 What does the unstable distribution contain?

The ”unstable” directory contains a snapshot of the 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 GNU/Linux software industry, but if it breaks: you get to keep both parts :-)

There are also main, contrib and non-free subdirectories in ”unstable”, separated on the same criteria as in ”stable”.

6.7 What are all those directories at the Debian archives?

The software that has been packaged for Debian GNU/Linux is available in one of several directory trees on each Debian mirror site.

The `dists` directory is short for "distributions", and it is the canonical way to access the currently available Debian releases (and pre-releases).

The `pool` directory contains the actual packages, see Section 6.10.

There are the following supplementary directories:

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

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

/indices/: various indices of the site (the Maintainers file and the override files).

/project/: mostly developer-only materials and some miscellaneous files.

6.8 What are all those directories inside `dists/stable/main`?

Within each of the major directory trees[2], there are three sets of subdirectories containing index files.

There's one set of `binary-`*`something`* subdirectories which contain index files for binary packages of each available computer architecture, for example `binary-i386` for packages which execute on Intel x86 PC machines or `binary-sparc` for packages which execute on Sun SPARCStations.

The complete list of available architectures for each release is available at the release's web page (`https://www.debian.org/releases`). For the current release, please see Section 4.1.

The index files in binary-* are called Packages(.gz, .bz2) and they include a summary of each binary package that is included in that distribution. The actual binary packages reside in the top level pool directory.

Furthermore, there's a subdirectory called source/ which contains index files for source packages included in the distribution. The index file is called Sources(.gz, .bz2).

Last but not least, there's a set of subdirectories meant for the installation system index files, they are at `debian-installer/binary-`*`architecture`*.

6.9 Where is 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.

The source code is distributed in the `pool` directory (see Section 6.10) together with all the architecture-specific binary directories. To retrieve the source code without having to be familiar with the structure of the archive, try a command like `apt-get source mypackagename`.

Due to restrictions in their licenses, source code may or may not be available for packages in the "contrib" and "non-free" areas, which are not formally part of the Debian system. In some cases only sourceless "binary blobs" can be distributed (see for instance `firmware-misc-nonfree`); in other cases the license prohibits the distribution of prebuilt binaries, but does allow packages of source code which users can compile locally (see `broadcom-sta-dkms`).

6.10 What's in the `pool` directory?

Packages are 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/`.

[2] `dists/stable/main`, `dists/stable/contrib`, `dists/stable/non-free`, and `dists/unstable/main/`, etc.

Additionally, since there are so many `lib*` packages, these are treated specially: for instance, libpaper packages are stored in `pool/main/libp/libpaper/`.
[3]

6.11 What is ”incoming”?

After a developer uploads a package, it stays for a short while in the ”incoming” directory before it is checked that it's genuine and allowed into the archive.

Usually nobody should install things from this place. However, in some rare cases of emergency, the incoming directory is available at https://incoming.debian.org/. You can manually fetch packages, check the GPG signature and MD5sums in the .changes and .dsc files, and then install them.

6.12 Where can I find old packages of previous releases?

Old releases are removed from the main archive and mirrors, which only keep the content of the releases up to ”oldstable” (the stable release before the current one). If you are interested in obtaining older versions of packages, go to https://snapshot.debian.org/.

The snapshot archive is a wayback machine that allows access to old packages based on dates and version numbers. It consists of all past and current packages the Debian archive provides. It provides a valuable resource for tracking down when regressions were introduced, or for providing a specific environment that a particular application may require to run. The snapshot archive is accessible like any normal apt repository, allowing it to be easily used by all.

6.13 How do I set up my own apt-able repository?

If you have built some private Debian packages which you'd like to install using the standard Debian package management tools, you can set up your own apt-able package archive. This is also useful if you'd like to share your Debian packages while these are not distributed by the Debian project. Instructions on how to do this are given on the Debian Wiki (`https://wiki.debian.org/HowToSetupADebianRepository`).

[3] Historically, packages were kept in the subdirectory of `dists` corresponding to which distribution contained them. This turned out to cause various problems, such as large bandwidth consumption on mirrors when major changes were made. This was fixed with the introduction of the package pool.

The `dists` directories are still used for the index files used by programs like `apt`.

Chapter 7

Basics of the Debian package management system

This chapter touches on some lower level internals of Debian package management. If you're interested mainly in *usage* of the relevant tools, skip to chapters Chapter 8 and/or Chapter 9.

7.1 What is a Debian package?

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 Section 7.2); they are usually characterized by having a '.deb' file extension. Binary packages can be unpacked using the Debian utility `dpkg` (possibly via a frontend like **apt**); 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 `.debian.tar.xz` 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. (The program **apt-get** can be used as a frontend for `dpkg-source`.)

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).) See more in Section 7.9 below.

Debian's packaging tools can be used to:

- manipulate and manage packages or parts of packages,
- administer local overrides of files in a package,
- aid developers in the construction of package archives, and
- aid users in the installation of packages which reside on a remote archive site.

7.2 What is the format of a Debian binary package?

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

The internals of this Debian binary packages format are described in the deb(5) manual page. This internal format is subject to change (between major releases of Debian GNU/Linux), therefore please always use dpkg-deb(1) if you need to do lowlevel manipulations on `.deb` files.

7.3 Why are Debian package file names so long?

The Debian binary package file names conform to the following convention: <DebianPackageName>_<VersionNumber>-<DebianRevisionNumber>_<DebianArchitecture>.deb

Checking the package name associated with a particular Debian archive file (.deb file) can be done 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 PPP_VVV-RRR_AAA.deb` (where PPP, VVV, RRR and AAA are the package name, version, revision and architecture 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 in place here, so the version number 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 a user who chooses to rebuild the package locally). 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.

The `AAA` component identifies the processor for which the package was built. This is commonly `amd64`, which refers to AMD64, Intel 64 or VIA Nano chips. For other possibilities review Debian's archive directory structure at Section 6.7. For details, see the description of "Debian architecture" in the manual page dpkg-architecture(1).

7.4 What is a Debian control file?

Specifics regarding the contents of a Debian control file are provided in the Debian Policy Manual, section 5, see Section 12.1.

Briefly, a sample control file is shown below for the Debian package hello:

```
Package: hello
Version: 2.9-2+deb8u1
Architecture: amd64
Maintainer: Santiago Vila <sanvila@debian.org>
Installed-Size: 145
Depends: libc6 (>= 2.14)
Conflicts: hello-traditional
Breaks: hello-debhelper (<< 2.9)
Replaces: hello-debhelper (<< 2.9), hello-traditional
Section: devel
Priority: optional
Homepage: https://www.gnu.org/software/hello/
Description: example package based on GNU hello
 The GNU hello program produces a familiar, friendly greeting.  It
 allows non-programmers to use a classic computer science tool which
 would otherwise be unavailable to them.
 .
 Seriously, though: this is an example of how to do a Debian package.
 It is the Debian version of the GNU Project's "hello world" program
 (which is itself an example for the GNU Project).
```

The Package field gives the package name. This is the name by which the package can be manipulated by the package tools, and usually similar to but not necessarily the same as the first component string in the Debian archive file name.

The Version field gives both the upstream developer's version number and (in the last component) the revision level of the Debian package of this program as explained in Section 7.3.

The Architecture field specifies the chip for which this particular binary was compiled.

The Depends field gives a list of packages that have to be installed in order to install this package successfully.

The Installed-Size indicates how much disk space the installed package will consume. This is intended to be used by installation front-ends in order to show whether there is enough disk space available to install the program.

The Section line gives the "section" where this Debian package is stored at the Debian archive sites.

The Priority indicates how important is this package for installation, so that semi-intelligent software like apt or aptitude can sort the package into a category of e.g. packages optionally installed. See Section 7.7.

The Maintainer field gives the e-mail address of the person who is currently responsible for maintaining this package.

The Description field gives a brief summary of the package's features.

For more information about all possible fields a package can have, please see the Debian Policy Manual, section 5, "Control files and their fields", see Section 12.1.

7.5 What is a Debian conffile?

Conffiles is a list of configuration files (usually placed in `/etc`) that the package management system will not overwrite when the package is upgraded. This ensures that local values for the contents of these files will be preserved, and is a critical feature enabling the in-place upgrade of packages on a running system.

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

```
dpkg --status package
```

And look under "Conffiles:".

7.6 What is a Debian preinst, postinst, prerm, and postrm script?

These files are executable scripts which are automatically run before or after a package is installed or removed. 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 is executed before the package it belongs to 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 the package `foo` once `foo` has been unpacked from its Debian archive (".deb") file. Many 'postinst' scripts 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 `foo`, and/or removes files created by the package. (Also see Section 7.8.)

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; you should not rely on it.)

7.7 What is an *Essential*, *Required*, *Important*, *Standard*, *Optional*, or *Extra* package?

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

- **Required**: packages that 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 probably not even be able to use dpkg to put things back. Systems with only the Required packages are probably unusable, 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 which the system will not run well or be usable without will be here. This does *NOT* include Emacs or X or TeX or any other large application. 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. Tools are included to be able to send e-mail (with mutt) and download files from archive servers.

  This is what will be installed by default if users do not select anything else. It does not include many large applications, but it does include the Python interpreter and some server software like OpenSSH (for remote administration) and Exim (for mail delivery, although it can be configured for local delivery only). It also includes some common generic documentation that most users will find helpful.

- **Optional** packages include all those that you might reasonably want to install if you do not know what they are, or that do not have specialized requirements.

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

- **Extra**: packages that 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”.

If you do a default Debian installation all the packages of priority **Standard** or higher will be installed in your system. If you select pre-defined tasks you will get lower priority packages too.

Additionally, some packages are marked as **Essential** since they are absolutely necessary for the proper functioning of the system. The package management tools will refuse to remove these.

7.8 What is a Virtual Package?

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 `konqueror` and `firefox-esr` programs are web browsers, and should therefore satisfy any dependency of a program that requires a web browser on a system, in order to work or to be useful. They are therefore both said to provide the ”virtual package” called `www-browser`.

Similarly, `exim4` 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 presence of this virtual package.

Debian provides a mechanism so that, if more than one package which provide the same virtual package is installed on a system, then system administrators can set one as the preferred package. The relevant command is `update-alternatives`, and is described further in Section 11.11.

7.9 What is meant by saying that a package *Depends*, *Recommends*, *Suggests*, *Conflicts*, *Replaces*, *Breaks* or *Provides* another package?

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 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" are 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 *breaks* Package B when both packages cannot be simultaneously configured in a system. The package management system will refuse to install one if the other one is already installed and configured in the system.
- 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 of these terms can be found in the Debian Policy manual, section 7.2, "Binary Dependencies", see Section 12.1.

7.10 What is meant by Pre-Depends?

"Pre-Depends" is a special dependency. In the case of most packages, `dpkg` will unpack the archive file of a package (i.e., its `.deb` file) independently of whether or not the files on which it depends exist on the system. Simplistically, unpacking 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, for some packages, `dpkg` will refuse even to unpack them until certain dependencies are resolved. Such packages are said to "Pre-depend" on the presence of some other packages. 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. the packages with the required priority and their LibC dependency.

As before, more detailed information about this can be found in the Policy manual.

7.11 What is meant by *unknown*, *install*, *remove*, *purge* and *hold* in the package status?

These "want" flags tell what the user wanted to do with a package (as indicated by the user's direct invocations of `dpkg`/`apt`/ `aptitude`).

Their meanings are:

- unknown - the user has never indicated whether the package is wanted.
- 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 file.
- 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. wants to keep the current version with the current status whatever that is.

7.12 How do I put a package on hold?

There are three ways of holding back packages, with dpkg, apt or aptitude.

With dpkg, you have to export the list of package selections, with:

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

Then edit the resulting file `selections.txt`, change 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
```

With apt, you can set a package to hold using

```
apt-mark hold package_name
```

and remove the hold with

```
apt-mark unhold package_name
```

With aptitude, you can hold a package using

```
aptitude hold package_name
```

and remove the hold with

```
aptitude unhold package_name
```

7.13 How do I install a source package?

Debian source packages can't actually be "installed", they are just unpacked in whatever directory you want to build the binary packages they produce.

Source packages are distributed on most of the same mirrors where you can obtain the binary packages. If you set up your APT's sources.list(5) to include the appropriate "deb-src" lines, you'll be able to easily download any source package by running

```
apt-get source foo
```

To help you in actually building the source package, Debian source packages provide the so-called build-dependencies mechanism. This means that the source package maintainer keeps a list of other packages that are required to build their package. To see how this is useful, run

```
apt-get build-dep foo
```

before building the source.

7.14 How do I build binary packages from a source package?

The preferred way to do this is by using various wrapper tools. We'll show how it's done using the `devscripts` tools. Install this package if you haven't done so already.

Now, first get the source package:

```
apt-get source foo
```

and change to the source tree:

```
cd foo-*
```

Then install needed build-dependencies (if any):

```
sudo apt-get build-dep foo
```

Then create a dedicated version of your own build (so that you won't get confused later when Debian itself releases a new version):

```
dch -l local 'Blah blah blah'
```

And finally build your package:

```
debuild -us -uc
```

If everything worked out fine, you should now be able to install your package by running

```
sudo dpkg -i ../*.deb
```

If you prefer to do things manually, and don't want to use `devscripts`, follow this procedure:

You will need all of foo_*.dsc, foo_*.tar.gz and foo_*.debian.tar.xz to compile the source (note: there is no .debian.tar.xz for some packages that are native to Debian).

Once you have them (Section 7.13) and if you have the `dpkg-dev` package installed, the following command:

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

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

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

```
dpkg-buildpackage -rfakeroot -b
```

to build the package (note that this also requires the `fakeroot` package), and then

```
dpkg -i ../foo_version-revision_arch.deb
```

to install the newly-built package(s).

7.15 How do I create Debian packages myself?

For a more detailed description on this, read the New Maintainers' Guide, available in the `maint-guide` package or at https://www.debian.org/doc/devel-manuals#maint-guide, or the Guide for Debian Maintainers, available in the `debmake-doc` package or at https://www.debian.org/doc/devel-manuals#debmake-doc.

Chapter 8

The Debian package management tools

8.1 What programs does Debian provide for managing its packages?

There are multiple tools that are used to manage Debian packages, from graphic or text-based interfaces to the low level tools used to install packages. All the available tools rely on the lower level tools to properly work and are presented here in decreasing complexity level.

It is important to understand that the higher level package management tools such as **aptitude** or **synaptic** rely on **apt** which, itself, relies on **dpkg** to manage the packages in the system.

See Chapter 2. Debian package management (`https://www.debian.org/doc/manuals/debian-referenc ch02.en.html`) of the Debian reference (`https://www.debian.org/doc/manuals/debian-reference/`) for more information about the Debian package management utilities. This document is available in various languages and formats, see the Debian Reference entry in the DDP Users' Manuals overview (`https://www.debian.org/doc/user-manuals#quick-reference`).

8.1.1 dpkg

This is the main package management program. **dpkg** can be invoked with many options. Some common uses are:

- Find out all the options: `dpkg --help`.
- Print out the control file (and other information) for a specified package: `dpkg --info foo_VVV-RRR.deb`.
- Install a package (including unpacking and configuring) onto the file system of the hard disk: `dpkg --install foo_VVV-RRR.deb`.
- Unpack (but do not configure) a Debian archive into the file system of the hard disk: `dpkg --unpack foo_VVV-RRR.deb`. Note that this operation does *not* necessarily leave the package in a usable state; some files may need further customization to run properly. This command removes any already-installed version of the program and runs the preinst (see Section 7.6) script associated with the package.
- Configure a package that already has been unpacked: `dpkg --configure foo`. Among other things, this action runs the postinst (see Section 7.6) script associated with the package. It also updates the files listed in the `conffiles` for this package. Notice that the 'configure' operation takes as its argument a package name (e.g., foo), *not* the name of a Debian archive file (e.g., foo_VVV-RRR.deb).
- Extract a single file named "blurf" (or a group of files named "blurf*") from a Debian archive: `dpkg --fsys-tarfile foo_VVV-RRR.deb | tar -xf - 'blurf*'`.
- Remove a package (but not its configuration files): `dpkg --remove foo`.

- Remove a package (including its configuration files): `dpkg --purge foo`.
- List the installation status of packages containing the string (or regular expression) "foo*": `dpkg --list 'foo*'`.

8.1.2 APT

APT is the *Advanced Package Tool*, an advanced interface to the Debian packaging system which provides the **apt-get** program. It provides commandline tools for searching and managing packages, and for querying information about them, as well as low-level access to all features of the libapt-pkg library. For more information, see the User's Guide in `/usr/share/doc/apt-doc/guide.html/index.html` (you will have to install the `apt-doc` package).

Starting with Debian Jessie, some frequently used **apt-get** and **apt-cache** commands have an equivalent via the new **apt** binary. This means some popular commands like **apt-get update**, **apt-get install**, **apt-get remove**, **apt-cache search**, or **apt-cache show** now can also be called simply via **apt**, say **apt update**, **apt install**, **apt remove**, **apt search**, or **apt show**. The following is an overview of the old and new commands:

```
apt-get update             ->  apt update
apt-get upgrade            ->  apt upgrade
apt-get dist-upgrade       ->  apt full-upgrade
apt-get install package    ->  apt install package
apt-get remove package     ->  apt remove package
apt-get autoremove         ->  apt autoremove
apt-cache search string    ->  apt search string
apt-cache policy package   ->  apt list -a package
apt-cache show package     ->  apt show package
apt-cache showpkg package  ->  apt show -a package
```

The **apt** tool merges functionality of apt-get and apt-cache and by default has a fancier colored output format, making it more pleasant for humans. For usage in scripts or advanced use cases, apt-get is still preferable or needed.

apt-get provides a simple way to retrieve and install packages from multiple sources using the command line. Unlike **dpkg**, **apt-get** does not understand .deb files, it works with the packages proper name and can only install .deb archives from a source specified in `/etc/apt/sources.list`. **apt-get** will call **dpkg** directly after downloading the .deb archives[1] from the configured sources.

Some common ways to use **apt-get** are:

- To update the list of packages known by your system, you can run:

  ```
  apt update
  ```

  (you should execute this regularly to update your package lists)

- To install the *foo* package and all its dependencies, run:

  ```
  apt install foo
  ```

- To remove the foo package from your system, run:

  ```
  apt remove foo
  ```

- To remove the foo package and its configuration files from your system, run:

  ```
  apt purge foo
  ```

- To list all packages for which newer versions are available, run:

  ```
  apt list --upgradable
  ```

[1] Notice that there are ports that make this tool available with other package management systems, like Red Hat package manager, also known as **rpm**

- To upgrade all the packages on your system (without installing extra packages or removing packages), run:

```
apt upgrade
```

- To upgrade all the packages on your system, and, if needed for a package upgrade, installing extra packages or removing packages, run:

```
apt full-upgrade
```

(The command `upgrade` keeps a package at its installed obsolete version if upgrading would need an extra package to be installed, for a new dependency to be satisfied. The `full-upgrade` command is less conservative.)

Note that you must be logged in as root to perform any commands that modify packages.

Note that **apt-get** now also installs recommended packages as default, and thanks to its robustness it's the preferred program for package management from console to perform system installation and major system upgrades.

The apt tool suite also includes the **apt-cache** tool to query the package lists. You can use it to find packages providing specific functionality through simple text or regular expression queries and through queries of dependencies in the package management system. Some common ways to use **apt-cache** are:

- To find packages whose description contain *word*:

```
apt search word
```

- To print the detailed information of a package:

```
apt show package
```

- To print the packages a given package depends on:

```
apt-cache depends package
```

- To print detailed information on the versions available for a package and the packages that reverse-depends on it:

```
apt-cache showpkg package
```

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

8.1.3 aptitude

aptitude is a package manager for Debian GNU/Linux systems that provides a frontend to the apt package management infrastructure. **aptitude** is a text-based interface using the curses library. Actions may be performed from a visual interface or from the command-line.

aptitude can be used to perform management tasks in a fast and easy way. It allows the user to view the list of packages and to perform package management tasks such as installing, upgrading, and removing packages.

aptitude provides the functionality of **apt-get**, as well as many additional features:

- **aptitude** offers easy access to all versions of a package.
- **aptitude** makes it easy to keep track of obsolete software by listing it under "Obsolete and Locally Created Packages".
- **aptitude** includes a fairly powerful system for searching particular packages and limiting the package display. Users familiar with **mutt** will pick up quickly, as **mutt** was the inspiration for the expression syntax.

- **aptitude** can be used to install the predefined tasks available. For more information see Section 8.1.5.
- **aptitude** in full screen mode has **su** functionality embedded and can be run by a normal user. It will call **su** (and ask for the root password, if any) when you really need administrative privileges.

You can use **aptitude** through a visual interface (simply run `aptitude`) or directly from the command line. The command line syntax used is very similar to the one used in **apt-get**. For example, to install the *foo* package, you can run `aptitude install foo`.

Note that **aptitude** is the preferred program for daily package management from the console.

For more information, read the manual page aptitude(8) and install the `aptitude-doc` package.

8.1.4 synaptic

synaptic is a graphical package manager. It enables you to install, upgrade and remove software packages in a user friendly way. Along with most of the features offered by aptitude, it also has a feature for editing the list of used repositories, and supports browsing all available documentation related to a package. See the Synaptic Website (`https://www.nongnu.org/synaptic/`) for more information.

8.1.5 tasksel

When you want to perform a specific task it might be difficult to find the appropriate suite of packages that fill your need. The Debian developers have defined `tasks`, a task is a collection of several individual Debian packages all related to a specific activity. Tasks can be installed through the **tasksel** program or through **aptitude**.

Typically, the Debian installer will automatically install the task associated with a standard system and a desktop environment. The specific desktop environment installed will depend on the CD/DVD media used, most commonly it will be the GNOME desktop (`gnome-desktop` task). Also, depending on your selections throughout the installation process, tasks might be automatically installed in your system. For example, if you selected a language other than English, the task associated with it will be installed automatically too.

8.1.6 Other package management tools

8.1.6.1 dpkg-deb

This program manipulates Debian archive (`.deb`) files. Some common uses are:

- Find out all the options: `dpkg-deb --help`.
- Determine what files are contained in a Debian archive file: `dpkg-deb --contents foo_VVV-RRR.deb`
- Extract the files contained in a named Debian archive into a user specified directory: `dpkg-deb --extract foo_VVV-RRR.deb tmp` extracts each of the files in `foo_VVV-RRR.deb` into the directory `tmp/`. This is convenient for examining the contents of a package in a localized directory, without installing the package into the root file system.
- Extract the control information files from a package: `dpkg-deb --control foo_VVV-RRR.deb tmp`.

Note that any packages that were merely unpacked using `dpkg-deb --extract` will be incorrectly installed, you should use `dpkg --install` instead.

More information is given in the manual page dpkg-deb(1).

8.2 Debian claims to be able to update a running program; how is this accomplished?

The kernel (file system) in Debian GNU/Linux 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 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 restart them as necessary.

8.3 How can I tell what packages are already installed on a Debian system?

To learn the status of all the packages installed on a Debian system, execute the command

```
dpkg --list
```

This prints out a one-line summary for each package, giving a 2-letter status symbol (explained in the header), the package name, the version which is *installed*, and a brief description.

To learn the status of packages whose names match any pattern beginning with "foo", run the command:

```
dpkg --list 'foo*'
```

To get a more verbose report for a particular package, execute the command:

```
dpkg --status packagename
```

8.4 How do I display the files of an installed package?

To list all the files provided by the installed package `foo` execute the command

```
dpkg --listfiles foo
```

Note that the files created by the installation scripts aren't displayed.

8.5 How can I find out what package produced a particular file?

To identify the package that produced the file named `foo` execute either:

- `dpkg --search foo`

  This searches for `foo` in installed packages. (This is (currently) equivalent to searching all of the files having the file extension of `.list` in the directory `/var/lib/dpkg/info/`, and adjusting the output to print the names of all the packages containing it, and diversions.)

  A faster alternative to this is the **dlocate** tool.

  ```
  dlocate -S foo
  ```

- `zgrep foo Contents-ARCH.gz`

  This searches for files which contain the substring `foo` in their full path names. The files `Contents-ARCH.gz` (where ARCH represents the wanted architecture) reside in the major package directories (main, non-free, contrib) at a Debian archive site (i.e. under `/debian/dists/trixie`). A `Contents` file refers only to the packages in the subdirectory tree where it resides. Therefore, a user might have to search more than one `Contents` files to find the package containing the file `foo`.

  This method has the advantage over `dpkg --search` in that it will find files in packages that are not currently installed on your system.

- `apt-file search` *`foo`*

  If you install the `apt-file` package, similar to the above, it searches files which contain the substring or regular expression `foo` in their full path names. The advantage over the example above is that there is no need to retrieve the `Contents-ARCH.gz` files as it will do this automatically for all the sources defined in `/etc/apt/sources.list` when you run (as root) `apt-file update`.

8.6 Why is ”foo-data” not removed when I uninstall ”foo”? How do I make sure old unused library-packages get purged?

Some packages are split in program (”foo”) and data (”foo-data”) (or in ”foo” and ”foo-doc”). This is true for many games, multimedia applications and dictionaries in Debian and has been introduced since some users might want to access the raw data without installing the program or because the program can be run without the data itself, making ”foo-data” optional.

Similar situations occur when dealing with libraries: generally these get installed since packages containing applications depend on them. When the application-package is purged, the library-package might stay on the system. Or: when the application-package no longer depends upon e.g. libdb4.2, but upon libdb4.3, the libdb4.2 package might stay when the application-package is upgraded.

In these cases, ”foo-data” doesn't depend on ”foo”, so when you remove the ”foo” package it will not get automatically removed by most package management tools. The same holds true for the library packages. This is necessary to avoid circular dependencies. However, if you use **apt-get** (see Section 8.1.2) or **aptitude** (see Section 8.1.3) as your package management tool, they will track automatically installed packages and give the possibility to remove them, when no packages making use of them remain in your system.

Chapter 9

Keeping your Debian system up-to-date

One of Debian's goals is to provide a consistent upgrade path and a secure upgrade process. We always do our best to make upgrading to new releases a smooth procedure. In case there's some important note to add to the upgrade process, the packages will alert the user, and often provide a solution to a possible problem.

You should also read the Release Notes document that describes the details of specific upgrades. It is available on the Debian website at https://www.debian.org/releases/stable/releasenotes and is also shipped on the Debian CDs, DVDs and Blu-ray discs.

9.1 How can I keep my Debian system current?

One could simply visit a Debian archive site, then peruse the directories until one finds the desired file, and then fetch it, and finally install it using `dpkg`. Note that `dpkg` will install upgrade files in place, even on a running system. Sometimes, 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 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 a more automated method. Several different packages are available for this purpose:

9.1.1 aptitude

aptitude is the recommended package manager for Debian GNU/Linux systems, and is described in Section 8.1.3.

Before you can use **aptitude** to make an upgrade, you'll have to edit the `/etc/apt/sources.list` file to set it up. If you wish to upgrade to the latest stable version of Debian, you'll probably want to use a source like this one:

```
http://deb.debian.org/debian stable main contrib
```

The mirror https://deb.debian.org/ is backed by a content-delivery network and requests to it will be directed to the closest instance to you. If you have a faster Debian mirror close to you, you can replace deb.debian.org with that one. See the mirror list at https://www.debian.org/mirror/list for more information.

Or you can use the redirector service httpredir.debian.org which aims to solve the problem of choosing a Debian mirror. It uses the geographic location of the user and other information to choose the best mirror that can serve the files. To take advantage of it use a source like this one:

```
http://httpredir.debian.org/debian stable main contrib
```

More details on this can be found in the sources.list(5) manual page.

To update your system from the command line, run

```
aptitude update
```

followed by

```
aptitude full-upgrade
```

Answer any questions that might come up, and your system will be upgraded.

Note that **aptitude** is not the recommended tool for doing upgrades from one Debian GNU/Linux release to another. Use **apt-get** instead. For upgrades between releases you should read the Release Notes (`https://www.debian.org/releases/stable/releasenotes`). This document describes in detail the recommended steps for upgrades from previous releases as well as known issues you should consider before upgrading.

For details, see the manual page aptitude(8), and the file `/usr/share/aptitude/README`.

9.1.2 apt-get and apt-cdrom

An alternative to **aptitude** is **apt-get** which is an APT-based command-line tool (described previously in Section 8.1.2).

apt-get, the APT-based command-line tool for handling packages, provides a simple, safe way to install and upgrade packages.

To use **apt-get**, edit the `/etc/apt/sources.list` file to set it up, just as for Section 9.1.1.

Then run

```
apt-get update
```

followed by

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

Answer any questions that might come up, and your system will be upgraded. See also the apt-get(8) manual page, as well as Section 8.1.2.

If you want to use CDs/DVDs/BDs to install packages, you can use **apt-cdrom**. For details, please see the Release Notes, section "Adding APT sources from optical media".

Please note that when you get and install the packages, you'll still have them kept in your /var directory hierarchy. To keep your partition from overflowing, remember to delete extra files using `apt-get clean` and `apt-get autoclean`, or to move them someplace else (hint: use `apt-move`).

9.2 Must I go into single user mode in order to upgrade a package?

No. Packages can be upgraded in place, even in running systems. Debian has a `start-stop-daemon` program that is invoked to stop, then restart running process if necessary during a package upgrade.

9.3 Do I have to keep all those .deb archive files on my disk?

No. If you have downloaded the files to your disk then after you have installed the packages, you can remove them from your system, e.g. by running `aptitude clean`.

9.4 How can I keep a log of the packages I added to the system? I'd like to know when upgrades and removals have occurred and on which packages!

Passing the `--log`-option to **dpkg** makes **dpkg** log status change updates and actions. It logs both the **dpkg**-invokation (e.g.

```
2005-12-30 18:10:33 install hello 1.3.18 2.1.1-4
```

) and the results (e.g.

```
2005-12-30 18:10:35 status installed hello 2.1.1-4
```

) If you'd like to log all your **dpkg** invocations (even those done using frontends like **aptitude**), you could add

```
log /var/log/dpkg.log
```

to your `/etc/dpkg/dpkg.cfg`. Be sure the created logfile gets rotated periodically. If you're using **logrotate**, this can be achieved by creating a file `/etc/logrotate.d/dpkg` with the following lines

```
/var/log/dpkg {
  missingok
  notifempty
}
```

More details on **dpkg** logging can be found in the dpkg(1) manual page.

aptitude logs the package installations, removals, and upgrades that it intends to perform to `/var/log/aptitude`. Note that the *results* of those actions are not recorded in this file!

Another way to record your actions is to run your package management session within the script(1) program.

9.5 Can I automatically update the system?

Yes. You can use **cron-apt**; this tool updates the system at regular intervals using a cron job. By default it just updates the package list and downloads new packages, but without installing them.

Note: Automatic upgrade of packages is **NOT** recommended in *testing* or *unstable* systems as this might bring unexpected behaviour and remove packages without notice.

9.6 I have several machines; how can I download the updates only one time?

If you have more than one Debian machine on your network, it is useful to use **apt-cacher** to keep all of your Debian systems up-to-date.

apt-cacher reduces the bandwidth requirements of Debian mirrors by restricting the frequency of Packages, Releases and Sources file updates from the back end and only doing a single fetch for any file, independently of the actual request from the proxy. **apt-cacher** automatically builds a Debian HTTP mirror based on requests which pass through the proxy.

Of course, you can get the same benefit if you are already using a standard caching proxy and all your systems are configured to use it.

Chapter 10

Debian and the kernel

10.1 Can I install and compile a kernel without some Debian-specific tweaking?

Yes.

There's only one common catch: the Debian C libraries are built with the most recent *stable* releases of the **kernel** headers. If you happen to need to compile a program with kernel headers newer than the ones from the stable branch, then you should either upgrade the package containing the headers (`linux-libc-dev`), or use the new headers from an unpacked tree of the newer kernel. That is, if the kernel sources are in `/usr/src/linux`, then you should add `-I/usr/src/linux/include/` to your command line when compiling.

10.2 What tools does Debian provide to build custom kernels?

Users who wish to (or must) build a custom kernel are encouraged to use the Debian package target included with recent versions of the kernel build system. After configuring the kernel, simply run the following command:

```
make deb-pkg
```

The new kernel package will be created in the directory one level above the kernel source tree, and it may be installed using `dpkg -i`.

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 `linux-source-`*`version`* package is available (where *`version`* stands for the kernel version).

10.3 What special provisions does Debian provide to deal with modules?

A configuration file containing modules to be manually loaded at boot time is kept at `/etc/modules`. However, editing this file is rarely needed.

Other module configuration is kept in the `/etc/modprobe.d/` directory. More information about the format of those files can be found in the modprobe.conf(5) manual page.

10.4 Can I safely de-install an old kernel package, and if so, how?

Yes. The **linux-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 remove unwanted kernel image packages using this command:

```
dpkg --purge linux-image-NNN
```

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

10.5 Where can I get more information about Linux packages for Debian?

Further information is maintained in the Debian Linux Kernel Handbook (`https://kernel-team.pages.debian.net/kernel-handbook/`).

Chapter 11

Customizing your Debian GNU/Linux system

11.1 How can I ensure that all programs use the same paper size?

Install the `libpaper1` package, and it will ask you for a system-wide default paper size. This setting will be kept in the file `/etc/papersize`.

Users can override the paper size setting using the `PAPERSIZE` environment variable. For details, see the manual page papersize(5).

11.2 How can I provide access to hardware peripherals, without compromising security?

Many device files in the `/dev` directory belong to some predefined groups. For example, `/dev/sr0` belongs to the `cdrom` group.

If you want a certain user to have access to one of these devices, just add the user to the group the device belongs to, i.e. do:

```
adduser user group
```

This way you won't have to change the file permissions on the device.

If you do this from within a user's shell or a GUI environment you have to logout and login again to become an effective member of that group. To check which groups you belong to run `groups`.

Notice that, since the introduction of `udev` if you change the permissions of a hardware peripheral, they might be adjusted for some devices when the system starts; if this happens to the hardware peripherals you are interested in, you will have to adjust the rules at `/etc/udev`.

11.3 How do I load a console font on startup the Debian way?

The `kbd` package supports this, edit the `/etc/kbd/config` file.

11.4 How can I configure an X11 program's application defaults?

Debian's X programs will install their application resource data in the `/etc/X11/app-defaults/` directory. If you want to customize X applications globally, put your customizations in those files. They are marked as configuration files, so their contents will be preserved during upgrades.

11.5 How does a Debian system boot?

Like all Unices, Debian boots up by executing the program `init`. Like most Linux distributions, a default Debian system uses `systemd` as the implementation of `init`. Traditional System-V style init and other

methods are also supported. [1]

To control the order in which services are started, traditional System-V style Unix systems use *run-levels*. These are replaced by *targets* under systemd. To display the default target to which systemd will bring the system, run the command

```
systemctl get-default
```

During boot-up, systemd starts the services or other targets listed in the default target file `/lib/systemd/system/defa`
The files for these services and targets are installed and the service is *enabled* during Debian package installation. If you specifically wish not to start a service during boot-up, instead of removing the corresponding package, you can run the command

```
systemctl disable service.service
```

using the name of the service file installed in `/lib/systemd/system` (usually based on the name of the package).

The *service file* `/lib/systemd/system/rc-local.service` provides an easy way to run customized scripts in the file `/etc/rc.local` after boot-up, similar to what's offered on Debian systems running System-V style init. Beware: this script will fail if it tries to interact with the console such as asking for a user password or trying to clear the screen.

You can check the status of any service by the command

```
service package status
```

. To start or stop a service, run

```
service package start
```

and

```
service package stop
```

. The `service` command works with any init system supported on a Debian system, not just with systemd. If you however prefer to use the same command on any systemd-supported Linux system, for checking the status run

```
systemctl status package.service
```

to get the same information.

For more information on systemd for Debian, see https://wiki.debian.org/systemd.

11.6 And how about Debian and traditional System V init?

Debian supports booting using traditional System V init, via the sysvinit-core package. The configuration file for System V `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 forking subprocesses to perform initialization such as to check and to mount file systems, to load modules, to start the network services, to set the clock, and to perform other initialization.

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] In 2014, Debian changed its default init system from System V init to systemd. Debian 8 "jessie" in April 2015 was the first release to ship with systemd as default init. Four decisions (https://www.debian.org/devel/tech-ctte#status) of the Debian Technical Committee were involved: Bug #727708 (https://lists.debian.org/20140211193904.GX24404@rzlab.ucr.edu) 2014-02-11: "The committee decided that the default init system for Linux architectures in jessie should be systemd." Bug #746715 (https://lists.debian.org/20140801023630.GF12356@teltox.donarmstrong.com) 2014-08-01: "The technical committee expects maintainers to continue to support the multiple available init systems", and merge reasonable contributions. Bug #746578 (https://lists.debian.org/20141116001628.GO32192@teltox.donarmstrong.com) 2014-11-15: "The committee decided that systemd-shim should be the first listed alternative dependency of libpam-systemd instead of systemd-sysv." This decision made it easier to keep running a non-systemd Debian system. Bug #762194 (https://lists.debian.org/21592.61064.527547.410074@chiark.greenend.org.uk)2017-11-04: "On automatic init system switching on upgrade"

- 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.

Debian uses dependency-based boot ordering through **insserv**, using the LSB headers in each script under `/etc/init.d/`, as well as parallel concurrent booting through the use of **startpar** to speed up the boot process.

The scripts in any of the directories, `/etc/rcN.d/` are just symbolic links back to scripts in `/etc/init.d/`. However, the *names* of the files in each of the `/etc/rcN.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.

Note that **invoke-rc.d** should not be used to call the `/etc/init.d/` scripts, **service** should be used instead.

11.7 And are there yet other ways of booting a Debian system?

If you do like System V init, but don't like the /etc/rc?.d/* links, you could install the `file-rc` package. That will convert the links into one single configuration file /etc/runlevel.conf instead.

If you like neither System V nor systemd, you might like `openrc` or `runit` or `daemontools`.

11.8 How does the package management system deal with packages that contain configuration files for other packages?

Some users wish to create, for example, a new server by installing a group of Debian packages and a locally generated package consisting of configuration files. This is not generally a good idea, because **dpkg** will not know about those configuration files if they are in a different package, and may write conflicting configurations when one of the initial "group" of packages is upgraded.

Instead, create a local package that modifies the configuration files of the "group" of Debian packages of interest. Then **dpkg** and the rest of the package management system will see that the files have been modified by the local "sysadmin" and will not try to overwrite them when those packages are upgraded.

11.9 How do I override a file installed by a package, so that a different version can be used instead?

Suppose a sysadmin or local user wishes to use a program "login-local" rather than the program "login" provided by the Debian `login` package.

Do **not**:

- Overwrite `/bin/login` with `login-local`.

The package management system will not know about this change, and will simply overwrite your custom `/bin/login` whenever `login` (or any package that provides `/bin/login`) is installed or updated.

Rather, do

- Execute:

```
dpkg-divert --divert /bin/login.debian /bin/login
```

  in order to cause all future installations of the Debian `login` package to write the file `/bin/login` to `/bin/login.debian` instead.

- Then execute:

```
cp login-local /bin/login
```

  to move your own locally-built program into place.

Run `dpkg-divert --list` to see which diversions are currently active on your system. Details are given in the manual page dpkg-divert(8).

11.10 How can I have my locally-built package included in the list of available packages that the package management system knows about?

Execute the command:

```
dpkg-scanpackages BIN_DIR OVERRIDE_FILE [PATHPREFIX] > my_Packages
```

where:

- BIN-DIR is a directory where Debian archive files (which usually have an extension of ".deb") are stored.
- OVERRIDE_FILE is a file that is edited by the distribution maintainers and is usually stored on a Debian archive at `indices/override.main.gz` for the Debian packages in the "main" distribution. You can ignore this for local packages.
- PATHPREFIX is an *optional* string that can be prepended to the `my_Packages` file being produced.

Once you have built the file `my_Packages`, tell the package management system about it by using the command:

```
dpkg --merge-avail my_Packages
```

If you are using APT, you can add the local repository to your sources.list(5) file, too.

11.11 Some users like mawk, others like gawk; some like vim, others like elvis; some like trn, others like tin; how does Debian support diversity?

There are several cases where two packages provide two different versions of a program, both of which provide the same core functionality. Users might prefer one over another out of habit, or because the user interface of one package is somehow more pleasing than the interface of another. Other users on the same system might make a different choice.

Debian uses a "virtual" package system to allow system administrators to choose (or let users choose) their favorite tools when there are two or more that provide the same basic functionality, yet satisfy package dependency requirements without specifying a particular package.

For example, there might exist two different versions of newsreaders on a system. The news server package might 'recommend' that there exist *some* news reader on the system, but the choice of `tin` or `trn` is left up to the individual user. This is satisfied by having both the `tin` and `trn` packages provide the virtual package `news-reader`. *Which* program is invoked is determined by a link pointing from a file with the virtual package name `/etc/alternatives/news-reader` to the selected file, e.g., `/usr/bin/trn`.

A single link is insufficient to support full use of an alternate program; normally, manual pages, and possibly other supporting files must be selected as well. The Perl script `update-alternatives` provides a way of ensuring that all the files associated with a specified package are selected as a system default.

For example, to check what executables provide "x-window-manager", run:

```
update-alternatives --display x-window-manager
```

If you want to change it, run:

```
update-alternatives --config x-window-manager
```

And follow the instructions on the screen (basically, press the number next to the entry you'd like better).

If a package doesn't register itself as a window manager for some reason (file a bug if it's in error), or if you use a window manager from /usr/local directory, the selections on screen won't contain your preferred entry. You can update the link through command line options, like this:

```
update-alternatives --install /usr/bin/x-window-manager \
  x-window-manager /usr/local/bin/wmaker-cvs 50
```

The first argument to "--install" option is the symlink that points to /etc/alternatives/NAME, where NAME is the second argument. The third argument is the program to which /etc/alternatives/NAME should point to, and the fourth argument is the priority (larger value means the alternative will more probably get picked automatically).

To remove an alternative you added, simply run:

```
update-alternatives --remove x-window-manager /usr/local/bin/wmaker-cvs
```

Chapter 12

Getting support for Debian GNU/Linux

12.1 What other documentation exists on and for a Debian system?

- Installation instructions for the current release: see https://www.debian.org/releases/stable/installmanual.

- The Debian GNU/Linux reference covers many aspects of system administration through shell-command examples. Basic tutorials, tips, and other information are provided for many different topics ranging from system administration to programming.

  Get it from the debian-reference package, or at https://www.debian.org/doc/user-manuals#quick-reference.

- The Debian Policy manual documents the policy requirements for the distribution, i.e. the structure and contents of the Debian archive, several design issues of the operating system etc. It also includes the technical requirements that each package must satisfy to be included in the distribution, and documents the basic technical aspects of Debian binary and source packages.

  Get it from the debian-policy package, or at https://www.debian.org/doc/devel-manuals#policy.

- Documentation developed by the Debian Documentation Project. It is available at https://www.debian.org/doc and includes user guides, administration guides and security guides for the Debian GNU/Linux operating system.

- Documentation on installed Debian packages: Most packages have files that are unpacked into /usr/share/doc/PACKAGE.

- Documentation on the Linux project: The Debian package doc-linux installs all of the most recent versions of the HOWTOs and mini-HOWTOs from the Linux Documentation Project (http://www.tldp.org/).

- Unix-style ”man” pages: Most commands have manual pages written in the style of the original Unix ”man” files. For instance, to see the manual page for the command ”ls”, execute man ls. Execute man man for more information on finding and viewing manual pages.

  New Debian users should note that the ’man’ pages of many general system commands are not available until they install these packages:

  - man-db, which contains the man program itself, and other programs for manipulating the manual pages.
  - manpages, which contains the system manual pages. (see Section 5.9).

- GNU-style "info" pages: User documentation for many commands, particularly GNU tools, is available not in "man" pages, but in "info" files which can be read by the GNU tool `info`, by running `M-x info` within GNU Emacs, or with some other Info page viewer.

  Its main advantage over the original "man" pages is that it is a hypertext system. It does *not* require the WWW, however; `info` can be run from a plain text console. It was designed by Richard Stallman and preceded the WWW.

Note that you may access a lot of documentation on your system by using a WWW browser, through `dwww`, `dhelp` or `doccentral` commands, found in respective packages, or by using `yelp`.

12.2 Are there any on-line resources for discussing Debian?

Yes. In fact, the main method of support Debian provides to our users is by the way of e-mail. We'll give some details on that, and mention some other useful resources. Even more resources are listed at the Debian Support webpage (`https://www.debian.org/support`).

12.2.1 Mailing lists

There are a lot of Debian-related mailing lists (`https://www.debian.org/MailingLists/`).

On a system with the `doc-debian` package installed there is a complete list of mailing lists in `/usr/share/doc/debian/mailing-lists.txt`.

Debian mailing lists are named following the pattern debian-*`list-subject`*. Examples are debian-announce, debian-user, debian-news. To subscribe to any list debian-*`list-subject`*, send mail to debian-*`list-subject`*-request@lists.debian.org with the word "subscribe" in the Subject: header. Be sure to remember to add *-request* to the e-mail address when using this method to subscribe or un-subscribe. Otherwise your e-mail will go to the list itself, which could be embarrassing or annoying, depending on your point of view.

You can subscribe to mailing lists using the WWW form (`https://www.debian.org/MailingLists/subscribe`). You can also un-subscribe using a WWW form (`https://www.debian.org/MailingLists/unsubscribe`).

The list manager's e-mail address is listmaster@lists.debian.org, in case you have any trouble.

The mailing lists are public forums. All e-mails sent to the lists are also copied to the public archive, for anybody (even non-subscribers) to browse or search. Please make sure you never send any confidential or unlicensed material to the lists. This includes things like e-mail addresses. Of particular note is the fact that spammers have been known to abuse e-mail addresses posted to our mailing lists. See the Mailing Lists Privacy policy (`https://www.debian.org/MailingLists/#disclaimer`) for more information.

Archives of the Debian mailing lists are available via WWW at `https://lists.debian.org/` which can be searched at Debian Mailing Lists Archives search (`https://lists.debian.org/search.html`).

12.2.1.1 What is the code of conduct for the mailing lists?

When using the Debian mailing lists, please follow these rules:

- Do not send spam. See the Debian mailing list advertising policy (`https://www.debian.org/MailingLists/#ads`).
- Do not flame; it is not polite. The people developing Debian are all volunteers, donating their time, energy and money in an attempt to bring the Debian project together.
- Do not use foul language; besides, some people receive the lists via packet radio, where swearing is illegal.
- Make sure that you are using the proper list. *Never* post your (un)subscription requests to the mailing list itself.[1]
- See section Section 12.5 for notes on reporting bugs.

[1] Use the debian-*`list-subject`*-REQUEST@lists.debian.org address for that.

12.2.2 Web forum

Debian User Forums (`http://forums.debian.net/`) provides web forums on which you can submit questions about Debian and have them answered by other users. (It is not an officially part of the Debian project.)

12.2.3 Wiki

Solutions to common problems, howtos, guides, tips and other documentation can be found at the constantly changing Debian Wiki (`https://wiki.debian.org/`).

12.2.4 Maintainers

Users can address questions to individual package maintainers using e-mail. To reach a maintainer of a package called xyz, send e-mail to *xyz@packages.debian.org*.

12.2.5 Usenet newsgroups

Users should post non-Debian-specific questions to one of the Linux USENET groups, which are named comp.os.linux.* or linux.*. There are several lists of Linux Usenet newsgroups and other related resources on the WWW, e.g. on the Linux Online (`https://www.linux.org/docs/usenet.html`) and LinuxJournal (`http://www.linuxjournal.com/helpdesk.php`) sites.

12.3 Is there a quick way to search for information on Debian GNU/Linux?

There is a variety of search engines that serve documentation related to Debian:

- Debian WWW search site (`https://search.debian.org/`).
- Debian Mailing Lists Archives search (`https://lists.debian.org/search.html`).

  For example, to find out what experiences people have had with finding drivers for NVIDIA graphic cards under Debian, try searching the phrase `NVIDIA Debian driver`. This will show you all the posts that contain these strings, i.e. those where people discussed these topics.
- Any of the common web spidering engines, such as DuckDuckGo (`https://duckduckgo.com/`) or Google (`https://www.google.com/`), as long as you use the right search terms.

  For example, searching on the string ”evince” gives a more detailed explanation of this package than the brief description field in its control file.

12.4 Are there logs of known bugs?

Reports on unsolved (and closed) issues are publicly available: Debian promised to do so by stating ”We will not hide problems” in the Debian Social Contract (`https://www.debian.org/social_contract`).

The Debian GNU/Linux distribution has a bug tracking system (BTS) which files details of bugs reported by users and developers. Each bug is given a number, and is kept on file. Once it has been dealt with, it is marked as such.

Copies of this information are available at `https://www.debian.org/Bugs/`.

A mail server provides access to the bug tracking system database via e-mail. In order to get the instructions, send an e-mail to request@bugs.debian.org with ”help” in the body.

12.5 How do I report a bug in Debian?

If you have found a bug in Debian, please read the instructions for reporting a bug in Debian. These instructions can be obtained in one of several ways:

- From the WWW. A copy of the instructions is shown at https://www.debian.org/Bugs/Reporting.
- On any Debian system with the doc-debian package installed. The instructions are in the file /usr/share/doc/debian/bug-reporting.txt.

You can use the package reportbug that will guide you through the reporting process and mail the message to the proper address, with some extra details about your system added automatically. It will also show you a list of bugs already reported to the package you are reporting against in case your bug has been reported previously, so that you can add additional information to the existing bug report.

Expect to get an automatic acknowledgement of your bug report. It will also be automatically given a bug tracking number, entered into the bug log and forwarded to the debian-bugs-dist mailing list.

Chapter 13

Contributing to the Debian Project

Donations (https://www.debian.org/donations) of time (to develop new packages, maintain existing packages, or provide user support), resources (to mirror the package and WWW archives), and money (to pay for new testbeds as well as hardware for the archives) can help the project. See also How can you help Debian? (https://www.debian.org/intro/help).

13.1 How can I become a Debian member/Debian developer?

The development of Debian is open to all, and new users with the right skills and/or the willingness to learn are needed to maintain existing packages which have been ”orphaned” by their previous maintainers, to develop new packages, to write documentation, to do translation work, to help with the Debian website, to provide user support, etc.

Volunteers can participate in the project as Debian contributor, Debian maintainer, or as a Debian developer (with or without uploading privileges):

- Debian contributors participate in many tasks, such as: writing documentation, maintaining the website, translating, supporting users, reviewing bug reports, or participating in different teams. It is an unofficial role and it is usually the starting point for many people.
- Debian maintainers maintain one or more specific packages and can upload these packages directly to the archive without requiring another Debian developer (or maintainer) to sponsor them.
- Debian developers is the full membership role in Debian. A developer can participate in Debian elections, can log into most of the systems Debian manages, and can upload any package to the archive. This role is provided after a strict and thorough process.

The website provides more information on how to join the project (https://www.debian.org/devel/join/). The description of becoming a Debian maintainer can be found at the New Member’s Corner (https://www.debian.org/devel/join/newmaint) at the Debian web site.

Debian uses Salsa (salsa.debian.org (https://salsa.debian.org/)), a GitLab instance, for collaborative development where anyone can sign up and send Merge Requests.

13.2 How can I contribute resources to the Debian project?

Since the project aims to make a substantial body of software rapidly and easily accessible throughout the globe, mirrors are needed. It is desirable but not absolutely necessary to mirror all of the archive. Please visit the Debian mirror size (https://www.debian.org/mirror/size) page for information on the disk space requirements.

Most of the mirroring is accomplished entirely automatically by scripts, without any interaction. However, the occasional glitch or system change occurs which requires human intervention.

If you have a high-speed connection to the Internet, the resources to mirror all or part of the distribution, and are willing to take the time (or find someone) who can provide regular maintenance of the system, then please contact debian-admin@lists.debian.org.

13.3 How can I contribute financially to the Debian project?

Donations from sponsors allow Debian to have machines, as well as other hardware, organise conferences and development sprints, amongst other things. For more information please visit Debian Donations (`https://www.debian.org/donations`). The page also lists the different methods that can be used to donate.

One can make individual donations to organizations that are critical to the development of the Debian project. The main organization is Software in the Public Interest, incorporated in the United States, but there are others.

13.3.1 Software in the Public Interest

Software in the Public Interest (SPI) is an IRS 501(c)(3) non-profit organization based in the United States. The purpose of the organization is to develop and distribute free software.

It encourages programmers to use the GNU General Public License or other licenses that allow free redistribution and use of software, and hardware developers to distribute documentation that will allow device drivers to be written for their product.

SPI acts as a fiscal sponsor to many free and open source projects. The Debian project has been an associate project since the organization's creation.

SPI can be reached at: `https://www.spi-inc.org/`.

13.3.2 Other organizations

There are a number of organizations created in different countries that hold assets in trust for Debian. The donations page (`https://www.debian.org/donations`) lists the trusted organizations individuals can donate to. At the time of this writing there are two of them: the Debian France Association (`https://france.debian.net/`) (in France), and debian.ch (`https://debian.ch/`) (Switzerland and the Principality of Liechtenstein). Additional affiliate organizations in other countries are listed in Organizations (`https://wiki.debian.org/Teams/Auditor/Organizations`) page in the Debian Wiki.

Chapter 14

Redistributing Debian GNU/Linux in a commercial product

14.1 Can I make and sell Debian CDs?

Go ahead. You do not need permission to distribute anything we have *released*, so that you can master your CD as soon as the beta-test ends. You do not have to pay us anything. Of course, all CD manufacturers must honor the licenses of the programs in Debian. For example, many of the programs are licensed under the GPL, which requires you to distribute their source code.

Also, we will publish a list of CD manufacturers who donate money, software, and time to the Debian project, and we will encourage users to buy from manufacturers who donate, so it is good advertising to make donations.

14.2 Can Debian be packaged with non-free software?

Yes. While all the main components of Debian are free software, we provide a non-free directory for programs that are not freely redistributable.

CD manufacturers *may* be able to distribute the programs we have placed in that directory, depending on the license terms or their private arrangements with the authors of those software packages. CD manufacturers can also distribute the non-free software they get from other sources on the same CD. This is nothing new: free and commercial software are distributed on the same CD by many manufacturers now. Of course we still encourage software authors to release the programs they write as free software.

14.3 I am making a special Linux distribution for a ”vertical market”. Can I use Debian GNU/Linux for the guts of a Linux system and add my own applications on top of it?

Yes. Debian-derived distributions are being created both in close cooperation with the Debian project itself and by external parties. One can use the Debian Pure Blends (`https://www.debian.org/blends/`) framework to work together with Debian; DebianEdu/Skolelinux (`https://wiki.debian.org/DebianEdu/`) is one such project.

There are several other Debian-derived distributions already on the market, such as grml, LMDE (Linux Mint Debian Edition), Knoppix and Ubuntu, that are targeted at a different kind of audience than the original Debian GNU/Linux is, but use most of our components in their product.

Debian also provides a mechanism to allow developers and system administrators to install local versions of selected files in such a way that they will not be overwritten when other packages are upgraded. This is discussed further in the question on Section 11.9.

14.4 Can I put my commercial program in a Debian ”package” so that it installs effortlessly on any Debian system?

Go right ahead. The package tool is free software; the packages may or may not be free software, it can install them all.

Chapter 15

Changes expected in the next major release of Debian

With each new release, the Debian project tries to focus on a set of topics. These are known as ”Release Goals” and they are all described in https://wiki.debian.org/ReleaseGoals/. Please note that the following sections might not be fully up-to-date, please refer to the Wiki for more information and the up-to-date status of these goals.

15.1 Hardening the system

It is a goal for the Debian project to ensure that any system installed is hardened and secure against attacks. There are several ways to achieve this, which include:

- Improve programs' security by compiling them with Security Hardening Build Flags (https://wiki.debian.org/ReleaseGoals/SecurityHardeningBuildFlags) in order to enable various protections against known security issues,
- Improve the default system configuration to make it less vulnerable to attacks (both local or remote),
- Enable security features delivered by new versions of the kernel.

All of these are done in an ongoing basis. For the first item, a set of security hardening build flags that try to prevent known attacks such as stack smashing, predictable locations of values in memory, etc. is used. The target is to cover at least all packages that are part of the basic installation as well as packages that had to be updated through a Security Advisory since 2006. As of this writing, around 400 packages have been modified since this effort was first started. All the issues are tracked in the BTS (https://bugs.debian.org/cgi-bin/pkgreport.cgi?tag=goal-hardening;users=hardening-discuss@lists.alioth.debian.org).

15.2 Extended support for non-English users

Debian already has very good support for non-English users, see Section 5.9.

We hope to find people who will provide support for even more languages, and translate programs and documents. Many programs and Debian-specific documents already support internationalization, so we need message catalogs translators. However, still some programs remain to be properly internationalized.

The GNU Translation Project ftp://ftp.gnu.org/pub/gnu/ABOUT-NLS works on internationalizing the GNU programs and different projects, such as the Desktop environments GNOME or KDE have their own translation teams. The goal of Debian is not to replace or to repeat the work done by these projects, indeed, Debian benefits from the work done by translators in these projects. However, there are still many programs which are not in the scope of those projects and which are translated within Debian.

Previous Debian releases have focused in topics such as:

- I18n support in all debconf-using packages: Packages using the Debian configuration management must allow for translation of all messages displayed to the user during package configuration.
- I18n support for package descriptions: Update package management frontends to use the translated descriptions of packages.
- UTF-8 debian/changelog and debian/control. This way, e.g. names of people from asian countries can get typeset the right way in changelogs.
- I18n support in the Debian Installer including full support for some languages that require the use of the graphical interface.

15.3 Improvements in the Debian Installer

Lots of work has been done on the Debian Installer, resulting in major improvements. We'll mention just two of them here.

Starting the installer from Microsoft Windows: It is now possible to start the installer directly from Microsoft Windows without the need to change BIOS settings. Upon insertion of a CD-ROM, DVD-ROM or USB stick, an autorun program will be started, offering a step-by-step process to start the Debian Installer.

15.4 More architectures

Complete Debian system on other architectures. Notice that even though some architectures are dropped for a given release, there still might be a way to install and upgrade using the latest `sid`.

15.5 More kernels

In addition to Debian GNU/Hurd, Debian is being ported also to BSD kernels, namely to FreeBSD (`https://www.debian.org/ports/kfreebsd-gnu/`). This port runs on both AMD64 ("kfreebsd-amd64") and traditional Intel ("kfreebsd-i386").

Chapter 16

General information about the FAQ

16.1 Authors

The first edition of this FAQ was made and maintained by J.H.M. Dassen (Ray) and Chuck Stickelman. Authors of the rewritten Debian GNU/Linux FAQ are Susan G. Kleinmann and Sven Rudolph. After them, the FAQ was maintained by Santiago Vila and, later, by Josip Rodin. The current maintainer is Javier Fernandez-Sanguino.

Parts of the information came from:

- The Debian-1.1 release announcement, by Bruce Perens (`https://perens.com/`),
- the Linux FAQ, by Ian Jackson (`https://www.chiark.greenend.org.uk/~ijackson/`),
- Debian Mailing Lists Archives (`https://lists.debian.org/`),
- the dpkg programmers' manual and the Debian Policy manual (see Section 12.1),
- many developers, volunteers, and beta testers,
- the flaky memories of its authors. :-)
- and the Choosing a Debian distribution FAQ (`http://KamarajuKusumanchi.github.io/choosing_debian_distribution/choosing_debian_distribution.html`), which Kamaraju Kusumanchi graciously released under the GPL, so it could be included as a new chapter (see Chapter 3).

The authors would like to thank all those who helped make this document possible.
All warranties are disclaimed. All trademarks are property of their respective trademark owners.

16.2 Feedback

Comments and additions to this document are always welcome. Please send e-mail to doc-debian@packages.debian.or
or submit a wishlist bug report against the `debian-faq` (`https://bugs.debian.org/debian-faq`) package.

16.3 Availability

The latest version of this document can be viewed on the Debian WWW pages at `https://www.debian.org/doc/FAQ/`.

It is also available for download in plain text, HTML, and PDF formats at `https://www.debian.org/doc/user-manuals#faq`. Also, there are several translations there.

This document is available in the `debian-faq` package. Translations are available in `debian-faq-de`, `debian-faq-fr` and other packages.

The original XML files used to create this document are also available in `debian-faq`'s source package, or on Salsa (salsa.debian.org), Debian's GitLab instance for collaborative development, at: `https://salsa.debian.org/ddp-team/debian-faq.git`. You can browse the repository at `https://salsa.debian.org/ddp-team/debian-faq`.

16.4 Document format

This document was written using the DocBook XML DTD. This 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.