Release-Notes for Debian 13 (trixie)

Debian Documentation Team

2025-06-27

CONTENTS

The Debian Documentation Project <https://www.debian.org/doc>.

Last updated on: 2025-06-27

CHAPTER

ONE

INTRODUCTION

This document informs users of the Debian distribution about major changes in version 13 (codenamed trixie).

The release notes provide information on how to upgrade safely from release 12 (codenamed bookworm) to the current release and inform users of known potential issues they could encounter in that process.

You can get the most recent version of this document from https://www.debian.org/releases/trixie/releasenotes.

> **Caution:** Note that it is impossible to list every known issue and that therefore a selection has been made based on a combination of the expected prevalence and impact of issues.

Please note that we only support and document upgrading from the previous release of Debian (in this case, the upgrade from bookworm). If you need to upgrade from older releases, we suggest you read previous editions of the release notes and upgrade to bookworm first.

1.1 Reporting bugs on this document

We have attempted to test all the different upgrade steps described in this document and to anticipate all the possible issues our users might encounter.

Nevertheless, if you think you have found a bug (incorrect information or information that is missing) in this documentation, please file a bug in the bug tracking system against the **release-notes** package. You might first want to review the existing bug reports in case the issue you've found has already been reported. Feel free to add additional information to existing bug reports if you can contribute content for this document.

We appreciate, and encourage, reports providing patches to the document's sources. You will find more information describing how to obtain the sources of this document in *Sources for this document*.

1.2 Contributing upgrade reports

We welcome any information from users related to upgrades from bookworm to trixie. If you are willing to share information please file a bug in the bug tracking system against the **upgrade-reports** package with your results. We request that you compress any attachments that are included (using `gzip`).

Please include the following information when submitting your upgrade report:

- The status of your package database before and after the upgrade: **dpkg**'s status database available at `/var/lib/dpkg/status` and **apt**'s package state information, available at `/var/lib/apt/extended_states`. You should have made a backup before the upgrade as described at *Back up any data or configuration information*, but you can also find backups of `/var/lib/dpkg/status` in `/var/backups`.

- Session logs created using `script`, as described in *Recording the session*.
- Your `apt` logs, available at `/var/log/apt/term.log`, or your `aptitude` logs, available at `/var/log/aptitude`.

Note: You should take some time to review and remove any sensitive and/or confidential information from the logs before including them in a bug report as the information will be published in a public database.

1.3 Sources for this document

The source of this document is in reStructuredText format, using the sphinx converter. The HTML version is generated using *sphinx-build -b html*. The PDF version is generated using *sphinx-build -b latex*. Sources for the Release Notes are available in the Git repository of the *Debian Documentation Project*. You can use the web interface to access its files individually through the web and see their changes. For more information on how to access Git please consult the Debian Documentation Project VCS information pages.

CHAPTER

TWO

WHAT'S NEW IN DEBIAN 13

The Wiki has more information about this topic.

2.1 Supported architectures

The following are the officially supported architectures for Debian 13:

- 64-bit PC (`amd64`)
- 64-bit ARM (`arm64`)
- ARM EABI (`armel`)
- ARMv7 (EABI hard-float ABI, `armhf`)
- 64-bit little-endian PowerPC (`ppc64el`)
- 64-bit little-endian RISC-V (`riscv64`)
- IBM System z (`s390x`)

Additionally, on 64-bit PC systems, a partial 32-bit userland (`i386`) is available. Please see *Reduced support for i386* for details.

You can read more about port status, and port-specific information for your architecture at the Debian port web pages.

2.2 What's new in the distribution?

This new release of Debian again comes with a lot more software than its predecessor bookworm; the distribution includes over 11294 new packages, for a total of over 59551 packages. Most of the software in the distribution has been updated: over 42821 software packages (this is 72% of all packages in bookworm). Also, a significant number of packages (over 9519, 16% of the packages in bookworm) have for various reasons been removed from the distribution. You will not see any updates for these packages and they will be marked as "obsolete" in package management front-ends; see *Obsolete packages*.

2.2.1 Official support for riscv64

This release for the first time officially supports the riscv64 architecture, allowing users to run Debian on 64-bit RISC-V hardware and benefit from all Debian 13 features.

The Wiki provides more details about riscv64 support in Debian.

2.2.2 PAC/BTI support on arm64

trixie introduces two security features on the arm64 architecture known as Pointer Authentication (PAC) and Branch Target Identification (BTI). They are designed to mitigate Return-Oriented Programming exploits and Jump-Oriented Programming attacks respectively.

The features are enabled automatically if your hardware supports them. The Wiki has information on how to check if your processor supports PAC/BTI and how they work.

2.2.3 Desktops and well known packages

Debian again ships with several desktop applications and environments. Among others it now includes the desktop environments GNOME 48, KDE Plasma 6.3, LXDE 13, LXQt 2.1.0, and Xfce 4.20.

Productivity applications have also been upgraded, including the office suites:

- LibreOffice is upgraded to version 25;
- GNUcash is upgraded to 5.10;

Among many others, this release also includes the following software updates:

Package	Version in 12 (bookworm)	Version in 13 (trixie)
Apache	2.4.62	2.4.63
Bash	5.2.15	5.2.37
BIND DNS Server	9.18	9.20
Cryptsetup	2.6	2.7
Emacs	28.2	30.1
Exim default e-mail server	4.96	4.98
GNU Compiler Collection as default compiler	12.2	14.2
GIMP	2.10.34	3.0.2
GnuPG	2.2.40	2.4.7
Inkscape	1.2.2	1.4
the GNU C library	2.36	2.41
Linux kernel image	6.1 series	6.12 series
LLVM/Clang toolchain	13.0.1 and 14.0 (default) and 15.0.6	19 (default), 17 and 18 available
MariaDB	10.11	11.8
Nginx	1.22	1.26
OpenJDK	17	21
OpenLDAP	2.5.13	2.6.9
OpenSSH	9.2p1	10.0p1
OpenSSL	3.0	3.4
Perl	5.36	5.40
PHP	8.2	8.4
Postfix MTA	3.7	3.10
PostgreSQL	15	17
Python 3	3.11	3.13
Rustc	1.63	1.85
Samba	4.17	4.22
Systemd	252	257
Vim	9.0	9.1

2.2.4 HTTP Boot Support

The Debian Installer and Debian Live Images can now be booted using "HTTP Boot" on supported UEFI and U-Boot firmware.

On systems using TianoCore firmware, enter the *Device Manager* menu, then choose *Network Device List*, select the network interface, *HTTP Boot Configuration*, and specify the full URL to the Debian ISO to boot.

For other firmware implementations, please see the documentation for your system's hardware and/or the firmware documentation.

CHAPTER

THREE

INSTALLATION SYSTEM

The Debian Installer is the official installation system for Debian. It offers a variety of installation methods. The methods that are available to install your system depend on its architecture.

Images of the installer for trixie can be found together with the Installation Guide on the Debian website (https://www.debian.org/releases/trixie/debian-installer/).

The Installation Guide is also included on the first media of the official Debian DVD (CD/blu-ray) sets, at:

```
/doc/install/manual/language/index.html
```

You may also want to check the errata for debian-installer at https://www.debian.org/releases/trixie/debian-installer#errata for a list of known issues.

3.1 What's new in the installation system?

There has been a lot of development on the Debian Installer since its previous official release with Debian 12, resulting in improved hardware support and some exciting new features or improvements.

If you are interested in an overview of the changes since bookworm, please check the release announcements for the trixie beta and RC releases available from the Debian Installer's news history.

3.2 Installing Debian Pure Blends

A selection of Debian Pure Blends, such as Debian Junior, Debian Science, or Debian FreedomBox, can now be accessed directly in the installer - see the installation-guide.

For information about Debian Pure Blends, visit https://www.debian.org/blends/ or the wiki.

3.3 Cloud installations

The cloud team publishes Debian trixie for several popular cloud computing services including:

- Amazon Web Services
- Microsoft Azure
- OpenStack
- Plain VM

Cloud images provide automation hooks via `cloud-init` and prioritize fast instance startup using specifically optimized kernel packages and grub configurations. Images supporting different architectures are provided where appropriate and the cloud team endeavors to support all features offered by the cloud service.

The cloud team will provide updated images until the end of the LTS period for trixie. New images are typically released for each point release and after security fixes for critical packages. The cloud team's full support policy can be found here.

More details are available at https://cloud.debian.org/ and on the wiki.

3.4 Container and Virtual Machine images

Multi-architecture Debian trixie container images are available on Docker Hub. In addition to the standard images, a "slim" variant is available that reduces disk usage.

Virtual machine images for the Hashicorp Vagrant VM manager are published to Vagrant Cloud.

CHAPTER

FOUR

UPGRADES FROM DEBIAN 12 (BOOKWORM)

4.1 Preparing for the upgrade

We suggest that before upgrading you also read the information in *Issues to be aware of for trixie*. That chapter covers potential issues which are not directly related to the upgrade process but could still be important to know about before you begin.

4.1.1 Back up any data or configuration information

Before upgrading your system, it is strongly recommended that you make a full backup, or at least back up any data or configuration information you can't afford to lose. The upgrade tools and process are quite reliable, but a hardware failure in the middle of an upgrade could result in a severely damaged system.

The main things you'll want to back up are the contents of `/etc`, `/var/lib/dpkg`, `/var/lib/apt/extended_states` and the output of:

```
$ dpkg --get-selections '*' # (the quotes are important)
```

If you use `aptitude` to manage packages on your system, you will also want to back up `/var/lib/aptitude/pkgstates`.

The upgrade process itself does not modify anything in the `/home` directory. However, some applications (e.g. parts of the Mozilla suite, and the GNOME and KDE desktop environments) are known to overwrite existing user settings with new defaults when a new version of the application is first started by a user. As a precaution, you may want to make a backup of the hidden files and directories ("dotfiles") in users' home directories. This backup may help to restore or recreate the old settings. You may also want to inform users about this.

Any package installation operation must be run with superuser privileges, so either log in as `root` or use `su` or `sudo` to gain the necessary access rights.

The upgrade has a few preconditions; you should check them before actually executing the upgrade.

4.1.2 Inform users in advance

It's wise to inform all users in advance of any upgrades you're planning, although users accessing your system via an `ssh` connection should notice little during the upgrade, and should be able to continue working.

If you wish to take extra precautions, back up or unmount the `/home` partition before upgrading.

You will have to do a kernel upgrade when upgrading to trixie, so a reboot will be necessary. Typically, this will be done after the upgrade is finished.

4.1.3 Prepare for downtime on services

There might be services that are offered by the system which are associated with packages that will be included in the upgrade. If this is the case, please note that, during the upgrade, these services will be stopped while their associated packages are being replaced and configured. During this time, these services will not be available.

The precise downtime for these services will vary depending on the number of packages being upgraded in the system, and it also includes the time the system administrator spends answering any configuration questions from package upgrades. Notice that if the upgrade process is left unattended and the system requests input during the upgrade there is a high possibility of services being unavailable[1] for a significant period of time.

If the system being upgraded provides critical services for your users or the network[2], you can reduce the downtime if you do a minimal system upgrade, as described in *Minimal system upgrade*, followed by a kernel upgrade and reboot, and then upgrade the packages associated with your critical services. Upgrade these packages prior to doing the full upgrade described in *Upgrading the system*. This way you can ensure that these critical services are running and available through the full upgrade process, and their downtime is reduced.

4.1.4 Prepare for recovery

Although Debian tries to ensure that your system stays bootable at all times, there is always a chance that you may experience problems rebooting your system after the upgrade. Known potential issues are documented in this and the next chapters of these Release Notes.

For this reason it makes sense to ensure that you will be able to recover if your system should fail to reboot or, for remotely managed systems, fail to bring up networking.

If you are upgrading remotely via an `ssh` link it is recommended that you take the necessary precautions to be able to access the server through a remote serial terminal. There is a chance that, after upgrading the kernel and rebooting, you will have to fix the system configuration through a local console. Also, if the system is rebooted accidentally in the middle of an upgrade there is a chance you will need to recover using a local console.

For emergency recovery we generally recommend using the *rescue mode* of the trixie Debian Installer. The advantage of using the installer is that you can choose between its many methods to find one that best suits your situation. For more information, please consult the section "Recovering a Broken System" in chapter 8 of the Installation Guide (at https://www.debian.org/releases/trixie/installmanual) and the Debian Installer FAQ.

If that fails, you will need an alternative way to boot your system so you can access and repair it. One option is to use a special rescue or live install image. After booting from that, you should be able to mount your root file system and `chroot` into it to investigate and fix the problem.

[1] If the debconf priority is set to a very high level you might prevent configuration prompts, but services that rely on default answers that are not applicable to your system will fail to start.

[2] For example: DNS or DHCP services, especially when there is no redundancy or failover. In the DHCP case end-users might be disconnected from the network if the lease time is lower than the time it takes for the upgrade process to complete.

Debug shell during boot using initrd

The **initramfs-tools** package includes a debug shell[3] in the initrds it generates. If for example the initrd is unable to mount your root file system, you will be dropped into this debug shell which has basic commands available to help trace the problem and possibly fix it.

Basic things to check are: presence of correct device files in `/dev`; what modules are loaded (`cat /proc/modules`); output of `dmesg` for errors loading drivers. The output of `dmesg` will also show what device files have been assigned to which disks; you should check that against the output of `echo $ROOT` to make sure that the root file system is on the expected device.

If you do manage to fix the problem, typing `exit` will quit the debug shell and continue the boot process at the point it failed. Of course you will also need to fix the underlying problem and regenerate the initrd so the next boot won't fail again.

Debug shell during boot using systemd

If the boot fails under systemd, it is possible to obtain a debug root shell by changing the kernel command line. If the basic boot succeeds, but some services fail to start, it may be useful to add `systemd.unit=rescue.target` to the kernel parameters.

Otherwise, the kernel parameter `systemd.unit=emergency.target` will provide you with a root shell at the earliest possible point. However, this is done before mounting the root file system with read-write permissions. You will have to do that manually with:

```
# mount -o remount,rw /
```

Another approach is to enable the systemd "early debug shell" via the `debug-shell.service`. On the next boot this service opens a root login shell on tty9 very early in the boot process. It can be enabled with the kernel boot parameter `systemd.debug-shell=1`, or made persistent with `systemctl enable debug-shell` (in which case it should be disabled again when debugging is completed).

More information on debugging a broken boot under systemd can be found in the Freedesktop.org Diagnosing Boot Problems article.

4.1.5 Prepare a safe environment for the upgrade

Important: If you are using some VPN services (such as **tinc**) consider that they might not be available throughout the upgrade process. Please see *Prepare for downtime on services*.

In order to gain extra safety margin when upgrading remotely, we suggest that you run upgrade processes in the virtual console provided by the `screen` program, which enables safe reconnection and ensures the upgrade process is not interrupted even if the remote connection process temporarily fails.

Users of the watchdog daemon provided by the **micro-evtd** package should stop the daemon and disable the watchdog timer before the upgrade, to avoid a spurious reboot in the middle of the upgrade process:

```
# service micro-evtd stop
# /usr/sbin/microapl -a system_set_watchdog off
```

[3] This feature can be disabled by adding the parameter `panic=0` to your boot parameters.

4.2 Start from "pure" Debian

The upgrade process described in this chapter has been designed for "pure" Debian stable systems. APT controls what is installed on your system. If your APT configuration mentions additional sources besides bookworm, or if you have installed packages from other releases or from third parties, then to ensure a reliable upgrade process you may wish to begin by removing these complicating factors.

The main configuration file that APT uses to decide what sources it should download packages from is `/etc/apt/sources.list`, but it can also use files in the `/etc/apt/sources.list.d/` directory - for details see sources.list(5). If your system is using multiple source-list files then you will need to ensure they stay consistent.

4.2.1 Upgrade to Debian 12 (bookworm)

Only upgrades from Debian 12 (bookworm) are supported. Display your Debian version with:

```
$ cat /etc/debian_version
```

Please follow the instructions in the Release Notes for Debian 12 at https://www.debian.org/releases/bookworm/releasenotes to upgrade to Debian 12 first if needed.

4.2.2 Upgrade to latest point release

This procedure assumes your system has been updated to the latest point release of bookworm. If you have not done this or are unsure, follow the instructions in *Upgrading your bookworm system*.

4.2.3 Debian Backports

Debian Backports allows users of Debian stable to run more up-to-date versions of packages (with some tradeoffs in testing and security support). The Debian Backports Team maintains a subset of packages from the next Debian release, adjusted and recompiled for usage on the current Debian stable release.

Packages from bookworm-backports have version numbers lower than the version in trixie, so they should upgrade normally to trixie in the same way as "pure" bookworm packages during the distribution upgrade. While there are no known potential issues, the upgrade paths from backports are less tested, and correspondingly incur more risk.

Caution: While regular Debian Backports are supported, there is no clean upgrade path from sloppy backports (which use APT source-list entries referencing bookworm-backports-sloppy).

As with *Unofficial sources*, users are advised to remove "bookworm-backports" entries from their APT source-list files before the upgrade. After it is completed, they may consider adding "trixie-backports" (see https://backports.debian.org/Instructions/).

For more information, consult the Backports Wiki page.

4.2.4 Prepare the package database

You should make sure the package database is ready before proceeding with the upgrade. If you are a user of another package manager like **aptitude** or **synaptic**, review any pending actions. A package scheduled for installation or removal might interfere with the upgrade procedure. Note that correcting this is only possible if your APT source-list files still point to "bookworm" and not to "stable" or "trixie"; see *Checking your APT source-list files*.

4.2.5 Remove obsolete packages

It is a good idea to *remove obsolete packages* from your system before upgrading. They may introduce complications during the upgrade process, and can present security risks as they are no longer maintained.

4.2.6 Remove non-Debian packages

Below there are two methods for finding installed packages that did not come from Debian, using either `apt` or `apt-forktracer`. Please note that neither of them are 100% accurate (e.g. the apt example will list packages that were once provided by Debian but no longer are, such as old kernel packages).

```
$ apt list '?narrow(?installed, ?not(?origin(Debian)))'
$ apt-forktracer | sort
```

4.2.7 Clean up leftover configuration files

A previous upgrade may have left unused copies of configuration files; *old versions* of configuration files, versions supplied by the package maintainers, etc. Removing leftover files from previous upgrades can avoid confusion. Find such leftover files with:

```
# find /etc -name '*.dpkg-*' -o -name '*.ucf-*' -o -name '*.merge-error'
```

4.2.8 The non-free and non-free-firmware components

If you have non-free firmware installed it is recommended to add `non-free-firmware` to your APT sources-list.

4.2.9 The proposed-updates section

If you have listed the `proposed-updates` section in your APT source-list files, you should remove it before attempting to upgrade your system. This is a precaution to reduce the likelihood of conflicts.

4.2.10 Unofficial sources

If you have any non-Debian packages on your system, you should be aware that these may be removed during the upgrade because of conflicting dependencies. If these packages were installed by adding an extra package archive in your APT source-list files, you should check if that archive also offers packages compiled for trixie and change the source item accordingly at the same time as your source items for Debian packages.

Some users may have *unofficial* backported "newer" versions of packages that *are* in Debian installed on their bookworm system. Such packages are most likely to cause problems during an upgrade as they may result in file conflicts[4]. *Possible issues during upgrade* has some information on how to deal with file conflicts if they should occur.

4.2.11 Disabling APT pinning

If you have configured APT to install certain packages from a distribution other than stable (e.g. from testing), you may have to change your APT pinning configuration (stored in `/etc/apt/preferences` and `/etc/apt/preferences.d/`) to allow the upgrade of packages to the versions in the new stable release. Further information on APT pinning can be found in apt_preferences(5).

4.2.12 Check package status

Regardless of the method used for upgrading, it is recommended that you check the status of all packages first, and verify that all packages are in an upgradable state. The following command will show any packages which have a status of Half-Installed or Failed-Config, and those with any error status.

```
$ dpkg --audit
```

You could also inspect the state of all packages on your system using `aptitude` or with commands such as

```
$ dpkg -l
```

or

```
# dpkg --get-selections '*' > ~/curr-pkgs.txt
```

Alternatively you can also use `apt`.

```
# apt list --installed > ~/curr-pkgs.txt
```

It is desirable to remove any holds before upgrading. If any package that is essential for the upgrade is on hold, the upgrade will fail.

```
$ apt-mark showhold
```

If you changed and recompiled a package locally, and didn't rename it or put an epoch in the version, you must put it on hold to prevent it from being upgraded.

The "hold" package state for `apt` can be changed using:

```
# apt-mark hold package_name
```

Replace `hold` with `unhold` to unset the "hold" state.

If there is anything you need to fix, it is best to make sure your APT source-list files still refer to bookworm as explained in *Checking your APT source-list files*.

[4] Debian's package management system normally does not allow a package to remove or replace a file owned by another package unless it has been defined to replace that package.

4.3 Preparing APT source-list files

Before starting the upgrade you must reconfigure APT source-list files (`/etc/apt/sources.list` and files under `/etc/apt/sources.list.d/`) to add sources for trixie and typically to remove sources for bookworm.

APT will consider all packages that can be found via any configured archive, and install the package with the highest version number, giving priority to the first entry in the files. Thus, if you have multiple mirror locations, list first the ones on local hard disks, then CD-ROMs, and then remote mirrors.

A release can often be referred to both by its codename (e.g. "bookworm", "trixie") and by its status name (i.e. "oldstable", "stable", "testing", "unstable"). Referring to a release by its codename has the advantage that you will never be surprised by a new release and for this reason is the approach taken here. It does of course mean that you will have to watch out for release announcements yourself. If you use the status name instead, you will just see loads of updates for packages available as soon as a release has happened.

Debian provides two announcement mailing lists to help you stay up to date on relevant information related to Debian releases:

- By subscribing to the Debian announcement mailing list, you will receive a notification every time Debian makes a new release. Such as when "trixie" changes from e.g. "testing" to "stable".
- By subscribing to the Debian security announcement mailing list, you will receive a notification every time Debian publishes a security announcement.

4.3.1 Adding APT Internet sources

On new installations the default is for APT to be set up to use the Debian APT CDN service, which should ensure that packages are automatically downloaded from a server near you in network terms. As this is a relatively new service, older installations may have configuration that still points to one of the main Debian Internet servers or one of the mirrors. If you haven't done so yet, it is recommended to switch over to the use of the CDN service in your APT configuration.

To make use of the CDN service, add a line like this to your APT source configuration (assuming you are using `main` and `contrib`):

```
deb https://deb.debian.org/debian trixie main contrib
```

After adding your new sources, disable the previously existing "`deb`" lines by placing a hash sign (`#`) in front of them.

However, if you get better results using a specific mirror that is close to you in network terms, this option is still available.

Debian mirror addresses can be found at https://www.debian.org/mirror/list.

For example, suppose your closest Debian mirror is `https://mirrors.kernel.org`. If you inspect that mirror with a web browser, you will notice that the main directories are organized like this:

```
https://mirrors.kernel.org/debian/dists/trixie/main/...
https://mirrors.kernel.org/debian/dists/trixie/contrib/...
```

To configure APT to use a given mirror, add a line like this (again, assuming you are using `main` and `contrib`):

```
deb https://mirrors.kernel.org/debian trixie main contrib
```

Note that the "`dists`" is added implicitly, and the arguments after the release name are used to expand the path into multiple directories.

Again, after adding your new sources, disable the previously existing archive entries.

4.3.2 Adding APT sources for a local mirror

Instead of using remote package mirrors, you may wish to modify the APT source-list files to use a mirror on a local disk (possibly mounted over NFS).

For example, your package mirror may be under `/var/local/debian/`, and have main directories like this:

```
/var/local/debian/dists/trixie/main/...
/var/local/debian/dists/trixie/contrib/...
```

To use this with **apt**, add this line to your `sources.list` file:

```
deb file:/var/local/debian trixie main contrib
```

Note that the "`dists`" is added implicitly, and the arguments after the release name are used to expand the path into multiple directories.

After adding your new sources, disable the previously existing archive entries in the APT source-list files by placing a hash sign (`#`) in front of them.

4.3.3 Adding APT sources from optical media

If you want to use *only* DVDs (or CDs or Blu-ray Discs), comment out the existing entries in all the APT source-list files by placing a hash sign (`#`) in front of them.

Make sure there is a line in `/etc/fstab` that enables mounting your CD-ROM drive at the `/media/cdrom` mount point. For example, if `/dev/sr0` is your CD-ROM drive, `/etc/fstab` should contain a line like:

```
/dev/sr0 /media/cdrom auto noauto,ro 0 0
```

Note that there must be *no spaces* between the words `noauto,ro` in the fourth field.

To verify it works, insert a CD and try running

```
# mount /media/cdrom    # this will mount the CD to the mount point
# ls -alF /media/cdrom  # this should show the CD's root directory
# umount /media/cdrom   # this will unmount the CD
```

Next, run:

```
# apt-cdrom add
```

for each Debian Binary CD-ROM you have, to add the data about each CD to APT's database.

4.4 Upgrading packages

The recommended way to upgrade from previous Debian releases is to use the package management tool `apt`.

Note: `apt` is meant for interactive use, and should not be used in scripts. In scripts one should use `apt-get`, which has a stable output better suitable for parsing.

Don't forget to mount all needed partitions (notably the root and `/usr` partitions) read-write, with a command like:

```
# mount -o remount,rw /mountpoint
```

Next you should double-check that the APT source entries (in `/etc/apt/sources.list` and files under `/etc/apt/sources.list.d/`) refer either to "trixie" or to "stable". There should not be any sources entries pointing to bookworm.

Note: Source lines for a CD-ROM might sometimes refer to "`unstable`"; although this may be confusing, you should *not* change it.

4.4.1 Recording the session

It is strongly recommended that you use the `/usr/bin/script` program to record a transcript of the upgrade session. Then if a problem occurs, you will have a log of what happened, and if needed, can provide exact information in a bug report. To start the recording, type:

```
# script -t 2>~/upgrade-trixie-step.time -a ~/upgrade-trixie-step.script
```

or similar. If you have to rerun the typescript (e.g. if you have to reboot the system) use different *step* values to indicate which step of the upgrade you are logging. Do not put the typescript file in a temporary directory such as `/tmp` or `/var/tmp` (files in those directories may be deleted during the upgrade or during any restart).

The typescript will also allow you to review information that has scrolled off-screen. If you are at the system's console, just switch to VT2 (using `Alt+F2`) and, after logging in, use

```
# less -R ~root/upgrade-trixie.script
```

to view the file.

After you have completed the upgrade, you can stop `script` by typing `exit` at the prompt.

`apt` will also log the changed package states in `/var/log/apt/history.log` and the terminal output in `/var/log/apt/term.log`. `dpkg` will, in addition, log all package state changes in `/var/log/dpkg.log`. If you use `aptitude`, it will also log state changes in `/var/log/aptitude`.

If you have used the *-t* switch for `script` you can use the `scriptreplay` program to replay the whole session:

```
# scriptreplay ~/upgrade-trixie-step.time ~/upgrade-trixie-step.script
```

4.4.2 Updating the package list

First the list of available packages for the new release needs to be fetched. This is done by executing:

```
# apt update
```

Note: Users of apt-secure may find issues when using `aptitude` or `apt-get`. For apt-get, you can use `apt-get update --allow-releaseinfo-change`.

4.4.3 Make sure you have sufficient space for the upgrade

You have to make sure before upgrading your system that you will have sufficient hard disk space when you start the full system upgrade described in *Upgrading the system*. First, any package needed for installation that is fetched from the network is stored in `/var/cache/apt/archives` (and the `partial/` subdirectory, during download), so you must make sure you have enough space on the file system partition that holds `/var/` to temporarily download the packages that will be installed in your system. After the download, you will probably need more space in other file system partitions in order to both install upgraded packages (which might contain bigger binaries or more data) and new packages that will be pulled in for the upgrade. If your system does not have sufficient space you might end up with an incomplete upgrade that is difficult to recover from.

`apt` can show you detailed information about the disk space needed for the installation. Before executing the upgrade, you can see this estimate by running:

```
# apt -o APT::Get::Trivial-Only=true full-upgrade
[ ... ]
XXX upgraded, XXX newly installed, XXX to remove and XXX not upgraded.
Need to get xx.xMB of archives.
After this operation, AAAMB of additional disk space will be used.
```

Note: Running this command at the beginning of the upgrade process may give an error, for the reasons described in the next sections. In that case you will need to wait until you've done the minimal system upgrade as in *Minimal system upgrade* before running this command to estimate the disk space.

If you do not have enough space for the upgrade, `apt` will warn you with a message like this:

```
E: You don't have enough free space in /var/cache/apt/archives/.
```

In this situation, make sure you free up space beforehand. You can:

- Remove packages that have been previously downloaded for installation (at `/var/cache/apt/archives`). Cleaning up the package cache by running `apt clean` will remove all previously downloaded package files.
- Remove forgotten packages. If you have used `aptitude` or `apt` to manually install packages in bookworm it will have kept track of those packages you manually installed, and will be able to mark as redundant those packages pulled in by dependencies alone which are no longer needed due to a package being removed. They will not mark for removal packages that you manually installed. To remove automatically installed packages that are no longer used, run:

  ```
  # apt autoremove
  ```

  You can also use `debfoster` to find redundant packages. Do not blindly remove the packages this tool presents, especially if you are using aggressive non-default options that are prone to false positives. It is highly recommended that you manually review the packages suggested for removal (i.e. their contents, sizes, and descriptions) before you remove them.
- Remove packages that take up too much space and are not currently needed (you can always reinstall them after the upgrade). If you have **popularity-contest** installed, you can use `popcon-largest-unused` to list the packages you do not use that occupy the most space. You can find the packages that just take up the most disk space with `dpigs` (available in the **debian-goodies** package) or with `wajig` (running `wajig size`). They can also be found with **aptitude**. Start `aptitude` in full-terminal mode, select `Views > New Flat Package List`, press `l` and enter `~i`, then press `S` and enter `~installsize`. This will give you a handy list to work with.
- Remove translations and localization files from the system if they are not needed. You can install the **localepurge** package and configure it so that only a few selected locales are kept in the system. This will reduce the disk space consumed at `/usr/share/locale`.

- Temporarily move to another system, or permanently remove, system logs residing under `/var/log/`.
- Use a temporary `/var/cache/apt/archives`: You can use a temporary cache directory from another filesystem (USB storage device, temporary hard disk, filesystem already in use, ...).

  Note: Do not use an NFS mount as the network connection could be interrupted during the upgrade.

  For example, if you have a USB drive mounted on `/media/usbkey`:

  1. remove the packages that have been previously downloaded for installation:

     ```
     # apt clean
     ```

  2. copy the directory `/var/cache/apt/archives` to the USB drive:

     ```
     # cp -ax /var/cache/apt/archives /media/usbkey/
     ```

  3. mount the temporary cache directory on the current one:

     ```
     # mount --bind /media/usbkey/archives /var/cache/apt/archives
     ```

  4. after the upgrade, restore the original `/var/cache/apt/archives` directory:

     ```
     # umount /var/cache/apt/archives
     ```

  5. remove the remaining `/media/usbkey/archives`.

  You can create the temporary cache directory on whatever filesystem is mounted on your system.

- Do a minimal upgrade of the system (see *Minimal system upgrade*) or partial upgrades of the system followed by a full upgrade. This will make it possible to upgrade the system partially, and allow you to clean the package cache before the full upgrade.

Note that in order to safely remove packages, it is advisable to switch your APT source-list files back to bookworm as described in *Checking your APT source-list files*.

4.4.4 Stop monitoring systems

As `apt` may need to temporarily stop services running on your computer, it's probably a good idea to stop monitoring services that can restart other terminated services during the upgrade. In Debian, **monit** is an example of such a service.

4.4.5 Minimal system upgrade

In some cases, doing the full upgrade (as described below) directly might remove large numbers of packages that you will want to keep. We therefore recommend a two-part upgrade process: first a minimal upgrade to overcome these conflicts, then a full upgrade as described in *Upgrading the system*.

To do this, first run:

```
# apt upgrade --without-new-pkgs
```

This has the effect of upgrading those packages which can be upgraded without requiring any other packages to be removed or installed.

The minimal system upgrade can also be useful when the system is tight on space and a full upgrade cannot be run due to space constraints.

If the **apt-listchanges** package is installed, it will (in its default configuration) show important information about upgraded packages in a pager after downloading the packages. Press `q` after reading to exit the pager and continue the upgrade.

4.4.6 Upgrading the system

Once you have taken the previous steps, you are now ready to continue with the main part of the upgrade. Execute:

```
# apt full-upgrade
```

This will perform a complete upgrade of the system, installing the newest available versions of all packages, and resolving all possible dependency changes between packages in different releases. If necessary, it will install some new packages (usually new library versions, or renamed packages), and remove any conflicting obsoleted packages.

When upgrading from a set of CDs/DVDs/BDs, you will probably be asked to insert specific discs at several points during the upgrade. You might have to insert the same disc multiple times; this is due to inter-related packages that have been spread out over the discs.

New versions of currently installed packages that cannot be upgraded without changing the install status of another package will be left at their current version (displayed as "held back"). This can be resolved by either using `aptitude` to choose these packages for installation or by trying `apt install package`.

4.5 Possible issues during upgrade

The following sections describe known issues that might appear during an upgrade to trixie.

4.5.1 Full-upgrade fails with "Could not perform immediate configuration"

In some cases the `apt full-upgrade` step can fail after downloading packages with:

```
E: Could not perform immediate configuration on 'package'.  Please see man 5 apt.conf␣
↪under APT::Immediate-Configure for details.
```

If that happens, running `apt full-upgrade -o APT::Immediate-Configure=0` instead should allow the upgrade to proceed.

Another possible workaround for this problem is to temporarily add both bookworm and trixie sources to your APT source-list files and run `apt update`.

4.5.2 Expected removals

The upgrade process to trixie might ask for the removal of packages on the system. The precise list of packages will vary depending on the set of packages that you have installed. These release notes give general advice on these removals, but if in doubt, it is recommended that you examine the package removals proposed by each method before proceeding. For more information about packages obsoleted in trixie, see *Obsolete packages*.

4.5.3 Conflicts or Pre-Depends loops

Sometimes it's necessary to enable the `APT::Force-LoopBreak` option in APT to be able to temporarily remove an essential package due to a Conflicts/Pre-Depends loop. `apt` will alert you of this and abort the upgrade. You can work around this by specifying the option `-o APT::Force-LoopBreak=1` on the `apt` command line.

It is possible that a system's dependency structure can be so corrupt as to require manual intervention. Usually this means using `apt` or

```
# dpkg --remove package_name
```

to eliminate some of the offending packages, or

```
# apt -f install
# dpkg --configure --pending
```

In extreme cases you might have to force re-installation with a command like

```
# dpkg --install /path/to/package_name.deb
```

4.5.4 File conflicts

File conflicts should not occur if you upgrade from a "pure" bookworm system, but can occur if you have unofficial backports installed. A file conflict will result in an error like:

```
Unpacking <package-foo> (from <package-foo-file>) ...
dpkg: error processing <package-foo> (--install):
trying to overwrite `<some-file-name>',
which is also in package <package-bar>
dpkg-deb: subprocess paste killed by signal (Broken pipe)
Errors were encountered while processing:
<package-foo>
```

You can try to solve a file conflict by forcibly removing the package mentioned on the *last* line of the error message:

```
# dpkg -r --force-depends package_name
```

After fixing things up, you should be able to resume the upgrade by repeating the previously described `apt` commands.

4.5.5 Configuration changes

During the upgrade, you will be asked questions regarding the configuration or re-configuration of several packages. When you are asked if any file in the `/etc/init.d` directory, or the `/etc/manpath.config` file should be replaced by the package maintainer's version, it's usually necessary to answer "yes" to ensure system consistency. You can always revert to the old versions, since they will be saved with a `.dpkg-old` extension.

If you're not sure what to do, write down the name of the package or file and sort things out at a later time. You can search in the typescript file to review the information that was on the screen during the upgrade.

4.5.6 Change of session to console

If you are running the upgrade using the system's local console you might find that at some points during the upgrade the console is shifted over to a different view and you lose visibility of the upgrade process. For example, this may happen in systems with a graphical interface when the display manager is restarted.

To recover the console where the upgrade was running you will have to use `Ctrl+Alt+F1` (if in the graphical startup screen) or `Alt+F1` (if in the local text-mode console) to switch back to the virtual terminal 1. Replace `F1` with the function key with the same number as the virtual terminal the upgrade was running in. You can also use `Alt+Left Arrow` or `Alt+Right Arrow` to switch between the different text-mode terminals.

4.6 Upgrading your kernel and related packages

This section explains how to upgrade your kernel and identifies potential issues related to this upgrade. You can either install one of the **linux-image-*** packages provided by Debian, or compile a customized kernel from source.

Note that a lot of information in this section is based on the assumption that you will be using one of the modular Debian kernels, together with **initramfs-tools** and **udev**. If you choose to use a custom kernel that does not require an initrd or if you use a different initrd generator, some of the information may not be relevant for you.

4.6.1 Installing a kernel metapackage

When you full-upgrade from bookworm to trixie, it is strongly recommended that you install a linux-image-* metapackage, if you have not done so before. These metapackages will automatically pull in a newer version of the kernel during upgrades. You can verify whether you have one installed by running:

```
$ dpkg -l 'linux-image*' | grep ^ii | grep -i meta
```

If you do not see any output, then you will either need to install a new linux-image package by hand or install a linux-image metapackage. To see a list of available linux-image metapackages, run:

```
$ apt-cache search linux-image- | grep -i meta | grep -v transition
```

If you are unsure about which package to select, run `uname -r` and look for a package with a similar name. For example, if you see "`4.9.0-8-amd64`", it is recommended that you install **linux-image-amd64**. You may also use `apt` to see a long description of each package in order to help choose the best one available. For example:

```
$ apt show linux-image-amd64
```

You should then use `apt install` to install it. Once this new kernel is installed you should reboot at the next available opportunity to get the benefits provided by the new kernel version. However, please have a look at *Things to do before rebooting* before performing the first reboot after the upgrade.

For the more adventurous there is an easy way to compile your own custom kernel on Debian. Install the kernel sources, provided in the **linux-source** package. You can make use of the `deb-pkg` target available in the sources' makefile for building a binary package. More information can be found in the Debian Linux Kernel Handbook, which can also be found as the **debian-kernel-handbook** package.

If possible, it is to your advantage to upgrade the kernel package separately from the main `full-upgrade` to reduce the chances of a temporarily non-bootable system. Note that this should only be done after the minimal upgrade process described in *Minimal system upgrade*.

4.7 Preparing for the next release

After the upgrade there are several things you can do to prepare for the next release.

- Remove newly redundant or obsolete packages as described in *Make sure you have sufficient space for the upgrade* and *Obsolete packages*. You should review which configuration files they use and consider purging the packages to remove their configuration files. See also *Purging removed packages*.

4.7.1 Purging removed packages

It is generally advisable to purge removed packages. This is especially true if these have been removed in an earlier release upgrade (e.g. from the upgrade to bookworm) or they were provided by third-party vendors. In particular, old init.d scripts have been known to cause issues.

Caution: Purging a package will generally also purge its log files, so you might want to back them up first.

The following command displays a list of all removed packages that may have configuration files left on the system (if any):

```
$ apt list '~c'
```

The packages can be removed by using `apt purge`. Assuming you want to purge all of them in one go, you can use the following command:

```
# apt purge '~c'
```

4.8 Obsolete packages

Introducing lots of new packages, trixie also retires and omits quite a few old packages that were in bookworm. It provides no upgrade path for these obsolete packages. While nothing prevents you from continuing to use an obsolete package where desired, the Debian project will usually discontinue security support for it a year after trixie's release[5], and will not normally provide other support in the meantime. Replacing them with available alternatives, if any, is recommended.

There are many reasons why packages might have been removed from the distribution: they are no longer maintained upstream; there is no longer a Debian Developer interested in maintaining the packages; the functionality they provide has been superseded by different software (or a new version); or they are no longer considered suitable for trixie due to bugs in them. In the latter case, packages might still be present in the "unstable" distribution.

"Obsolete and Locally Created Packages" can be listed and purged from the commandline with:

```
$ apt list '~o'
# apt purge '~o'
```

The Debian Bug Tracking System often provides additional information on why the package was removed. You should review both the archived bug reports for the package itself and the archived bug reports for the ftp.debian.org pseudo-package.

For a list of obsolete packages for trixie, please refer to *Noteworthy obsolete packages*.

[5] Or for as long as there is not another release in that time frame. Typically only two stable releases are supported at any given time.

4.8.1 Transitional dummy packages

Some packages from bookworm may have been replaced in trixie by transitional dummy packages, which are empty placeholders designed to simplify upgrades. If for instance an application that was formerly a single package has been split into several, a transitional package may be provided with the same name as the old package and with appropriate dependencies to cause the new ones to be installed. After this has happened the redundant dummy package can be safely removed.

The package descriptions for transitional dummy packages usually indicate their purpose. However, they are not uniform; in particular, some "dummy" packages are designed to be kept installed, in order to pull in a full software suite, or track the current latest version of some program.

CHAPTER

FIVE

ISSUES TO BE AWARE OF FOR TRIXIE

Sometimes, changes introduced in a new release have side-effects we cannot reasonably avoid, or they expose bugs somewhere else. This section documents issues we are aware of. Please also read the errata, the relevant packages' documentation, bug reports, and other information mentioned in *Further reading*.

5.1 Things to be aware of while upgrading to trixie

This section covers items related to the upgrade from bookworm to trixie.

5.1.1 Reduced support for i386

From trixie, i386 is no longer supported as a regular architecture: there is no official kernel and no Debian installer for i386 systems. Fewer packages are available for i386 because many projects no longer support it. The architecture's sole remaining purpose is to support running legacy code, for example, by way of multiarch or a chroot on a 64-bit (amd64) system.

The i386 architecture is now only intended to be used on a 64-bit (amd64) CPU. Its instruction set requirements include SSE2 support, so it will not run successfully on most of the 32-bit CPU types that were supported by Debian 12.

Users running i386 systems should not upgrade to trixie. Instead, Debian recommends either reinstalling them as amd64, where possible, or retiring the hardware. Cross-grading without a reinstall is a technically possible, but risky, alternative.

5.1.2 openssh-server no longer reads ~/.pam_environment

The Secure Shell (SSH) daemon provided in the **openssh-server** package, which allows logins from remote systems, no longer reads the user's `~/.pam_environment` file by default; this feature has a history of security problems and has been deprecated in current versions of the Pluggable Authentication Modules (PAM) library. If you used this feature, you should switch from setting variables in `~/.pam_environment` to setting them in your shell initialization files (e.g. `~/.bash_profile` or `~/.bashrc`) or some other similar mechanism instead.

Existing SSH connections will not be affected, but new connections may behave differently after the upgrade. If you are upgrading remotely, it is normally a good idea to ensure that you have some other way to log into the system before starting the upgrade; see *Prepare for recovery*.

5.1.3 OpenSSH no longer supports DSA keys

Digital Signature Algorithm (DSA) keys, as specified in the Secure Shell (SSH) protocol, are inherently weak: they are limited to 160-bit private keys and the SHA-1 digest. The SSH implementation provided by the **openssh-client** and **openssh-server** packages has disabled support for DSA keys by default since OpenSSH 7.0p1 in 2015, released with Debian 9 ("stretch"), although it could still be enabled using the `HostKeyAlgorithms` and `PubkeyAcceptedAlgorithms` configuration options for host and user keys respectively.

The only remaining uses of DSA at this point should be connecting to some very old devices. For all other purposes, the other key types supported by OpenSSH (RSA, ECDSA, and Ed25519) are superior.

As of OpenSSH 9.8p1 in trixie, DSA keys are no longer supported even with the above configuration options. If you have a device that you can only connect to using DSA, then you can use the `ssh1` command provided by the **openssh-client-ssh1** package to do so.

In the unlikely event that you are still using DSA keys to connect to a Debian server (if you are unsure, you can check by adding the `-v` option to the `ssh` command line you use to connect to that server and looking for the "Server accepts key:" line), then you must generate replacement keys before upgrading. For example, to generate a new Ed25519 key and enable logins to a server using it, run this on the client, replacing `username@server` with the appropriate user and host names:

```
$ ssh-keygen -t ed25519
$ ssh-copy-id username@server
```

5.1.4 The last, lastb and lastlog commands have been replaced

The **util-linux** package no longer provides the `last` or `lastb` commands, and the **login** package no longer provides `lastlog`. These commands provided information about previous login attempts using `/var/log/wtmp`, `/var/log/btmp`, `/var/run/utmp` and `/var/log/lastlog`, but these files will not be usable after 2038 because they do not allocate enough space to store the login time (the Year 2038 Problem), and the upstream developers do not want to change the file formats. Most users will not need to replace these commands with anything, but the **util-linux** package provides a `lslogins` command which can tell you when accounts were last used.

There are two direct replacements available: `last` can be replaced by `wtmpdb` from the **wtmpdb** package (the **libpam-wtmpdb** package also needs to be installed) and `lastlog` can be replaced by `lastlog2` from the **lastlog2** package (**libpam-lastlog2** also needs to be installed). If you want to use these, you will need to install the new packages after the upgrade, see the util-linux NEWS.Debian for further information. The command `lslogins --failed` provides similar information to `lastb`.

If you do not install **wtmpdb** then we recommend you remove old log files `/var/log/wtmp*`. If you do install **wtmpdb** it will upgrade `/var/log/wtmp` and you can read older wtmp files with `wtmpdb import -f <dest>`. There is no tool to read `/var/log/lastlog*` or `/var/log/btmp*` files: they can be deleted after the upgrade.

5.1.5 RabbitMQ no longer supports HA queues

High-availability (HA) queues are no longer supported by **rabbitmq-server** starting in trixie. To continue with an HA setup, these queues need to be switched to "quorum queues".

If you have an OpenStack deployment, please switch the queues to quorum before upgrading. Please also note that beginning with OpenStack's "Caracal" release in trixie, OpenStack supports only quorum queues.

5.1.6 RabbitMQ cannot be directly upgraded from bookworm

There is no direct, easy upgrade path for RabbitMQ from bookworm to trixie. Details about this issue can be found in bug 1100165.

The recommended upgrade path is to completely wipe the rabbitmq database and restart the service (after the trixie upgrade). This may be done by deleting `/var/lib/rabbitmq/mnesia` and all of its contents.

5.1.7 MariaDB major version upgrades only work reliably after a clean shutdown

MariaDB does not support error recovery across major versions. For example if a MariaDB 10.11 server experienced an abrupt shutdown due to power loss or software defect, the database needs to be restarted with the same MariaDB 10.11 binaries so it can do successful error recovery and reconcile the data files and log files to roll-forward or revert transactions that got interrupted.

If you attempt to do crash recovery with MariaDB 11.8 using the data directory from a crashed MariaDB 10.11 instance, the newer MariaDB server will refuse to start.

To ensure a MariaDB Server is shut down cleanly before going into major version upgrade, stop the service with

```
# service mariadb stop
```

followed by checking server logs for `Shutdown complete` to confirm that flushing all data and buffers to disk completed successfully.

If it didn't shut down cleanly, restart it to trigger crash recovery, wait, stop again and verify that second stop was clean.

For additional information about how to make backups and other relevant information for system administrators, please see /usr/share/doc/mariadb-server/README.Debian.gz.

5.1.8 Ping no longer runs with elevated privileges

The default version of ping (provided by **iputils-ping**) is no longer installed with access to the *CAP_NET_RAW* linux capability, but instead uses `ICMP_PROTO` datagram sockets for network communication. Access to these sockets is controlled based on the user's Unix group membership using the `net.ipv4.ping_group_range` sysctl. In normal installations, the **linux-sysctl-defaults** package will set this value to a broadly permissive value, allowing unprivileged users to use ping as expected, but some upgrade scenarios may not automatically install this package. See `/usr/lib/sysctl.d/50-default.conf` and the kernel documentation for more information on the semantics of this variable.

5.1.9 Dovecot configuration changes

The *dovecot* email server suite in trixie uses a configuration format that is incompatible with previous versions. Details about the configuration changes are available at docs.dovecot.org.

In order to avoid potentially extended downtime, you are strongly encouraged to port your configuration in a staging environment before beginning the upgrade of a production mail system.

5.1.10 Significant changes to libvirt packaging

The **libvirt-daemon** package, which provides an API and toolkit for managing virtualization platforms, has been overhauled in trixie. Each driver and storage backend now comes in a separate binary package, which enables much greater flexibility.

Care is taken during upgrades from bookworm to retain the existing set of components, but in some cases functionality might end up being temporarily lost. We recommend that you carefully review the list of installed binary packages after upgrading to ensure that all the expected ones are present; this is also a great time to consider uninstalling unwanted components.

In addition, some conffiles might end up marked as "obsolete" after the upgrade. The `/usr/share/doc/libvirt-common/NEWS.Debian.gz` file contains additional information on how to verify whether your system is affected by this issue and how to address it.

5.1.11 Things to do before rebooting

When `apt full-upgrade` has finished, the "formal" upgrade is complete. For the upgrade to trixie, there are no special actions needed before performing a reboot.

5.2 Items not limited to the upgrade process

5.2.1 The directories /tmp and /var/tmp are now regularly cleaned

On new installations, *systemd-tmpfiles* will now regularly delete old files in `/tmp` and `/var/tmp` while the system is running. This change makes Debian consistent with other distributions. Because there is a small risk of data loss, it has been made "opt-in": the upgrade to trixie will create a file /etc/tmpfiles.d/tmp.conf which reinstates the old behavior. This file can be deleted to adopt the new default, or edited to define custom rules. The rest of this section explains the new default and how to customize it.

The new default behavior is for files in `/tmp` to be automatically deleted after 10 days from the time they were last used (as well as after a reboot). Files in `/var/tmp` are deleted after 30 days (but not deleted after a reboot).

Before adopting the new default, you should either adapt any local programs that store data in `/tmp` or `/var/tmp` for long periods to use alternative locations, such as `~/tmp/`, or tell *systemd-tmpfiles* to exempt the data file from deletion by creating a file `local-tmp-files.conf` in `/etc/tmpfiles.d` containing lines such as:

```
x /var/tmp/my-precious-file.pdf
x /tmp/foo
```

Please see systemd-tmpfiles(8) and tmpfiles.d(5) for more information.

5.2.2 Limitations in security support

There are some packages where Debian cannot promise to provide minimal backports for security issues. These are covered in the following subsections.

Note: The package **debian-security-support** helps to track the security support status of installed packages.

Security status of web browsers and their rendering engines

Debian 13 includes several browser engines which are affected by a steady stream of security vulnerabilities. The high rate of vulnerabilities and partial lack of upstream support in the form of long term branches make it very difficult to support these browsers and engines with backported security fixes. Additionally, library interdependencies make it extremely difficult to update to newer upstream releases. Applications using the **webkit2gtk** source package (e.g. **epiphany**) are covered by security support, but applications using qtwebkit (source package **qtwebkit-opensource-src**) are not.

For general web browser use we recommend Firefox or Chromium. They will be kept up-to-date by rebuilding the current ESR releases for stable. The same strategy will be applied for Thunderbird.

Once a release becomes `oldstable`, officially supported browsers may not continue to receive updates for the standard period of coverage. For example, Chromium will only receive 6 months of security support in `oldstable` rather than the typical 12 months.

Go- and Rust-based packages

The Debian infrastructure currently has problems with rebuilding packages of types that systematically use static linking. With the growth of the Go and Rust ecosystems it means that these packages will be covered by limited security support until the infrastructure is improved to deal with them maintainably.

In most cases if updates are warranted for Go or Rust development libraries, they will only be released via regular point releases.

5.3 Obsolescence and deprecation

5.3.1 Noteworthy obsolete packages

The following is a list of known and noteworthy obsolete packages (see *Obsolete packages* for a description).

The list of obsolete packages includes:

- The **libnss-gw-name** package has been removed from trixie. The upstream developer suggests using **libnss-myhostname** instead.
- The **pcregrep** package has been removed from trixie. It can be replaced with `grep -P` (`--perl-regexp`) or `pcre2grep` (from **pcre2-utils**).

5.3.2 Deprecated components for trixie

With the next release of Debian 14 (codenamed forky) some features will be deprecated. Users will need to migrate to other alternatives to prevent trouble when updating to Debian 14.

This includes the following features:

- The **sudo-ldap** package will be removed in forky. The Debian sudo team has decided to discontinue it due to maintenance difficulties and limited use. New and existing systems should use **libsss-sudo** instead.

  Upgrading Debian trixie to forky without completing this migration may result in the loss of intended privilege escalation.

  For further details, please refer to bug 1033728 and to the NEWS file in the **sudo** package.

- The **sudo_logsrvd** feature, used for sudo input/output logging, may be removed in Debian forky unless a maintainer steps forward. This component is of limited use within the Debian context, and maintaining it adds unnecessary complexity to the basic sudo package.

  For ongoing discussions, see bug 1101451 and the NEWS file in the **sudo** package.

- The **libnss-docker** package is no longer developed upstream and requires version 1.21 of the Docker API. That deprecated API version is still supported by Docker Engine v26 (shipped by Debian trixie) but will be removed in Docker Engine v27+ (shipped by Debian forky). Unless upstream development resumes, the package will be removed in Debian forky.

- The **openssh-client** and **openssh-server** packages currently support GSS-API authentication and key exchange, which is usually used to authenticate to Kerberos services. This has caused some problems, especially on the server side where it adds new pre-authentication attack surface, and Debian's main OpenSSH packages will therefore stop supporting it starting with forky.

  If you are using GSS-API authentication or key exchange (look for options starting with `GSSAPI` in your OpenSSH configuration files) then you should install the **openssh-client-gssapi** (on clients) or **openssh-server-gssapi** (on servers) package now. On trixie, these are empty packages depending on **openssh-client** and **openssh-server** respectively; on forky, they will be built separately.

- sbuild-debian-developer-setup has been deprecated in favor of sbuild+unshare

  sbuild, the tool to build Debian packages in a minimal environment, has had a major upgrade and should work out of the box now. As a result the package **sbuild-debian-developer-setup** is no longer needed and has been deprecated. You can try the new version with:

```
$ sbuild --chroot-mode=unshare --dist=unstable hello
```

- The **fcitx** packages have been deprecated in favor of **fcitx5**

  The **fcitx** input method framework, also known as **fcitx4** or **fcitx 4.x**, is no longer maintained upstream. As a result, all related input method packages are now deprecated. The package **fcitx** and packages with names beginning with **fcitx-** will be removed in Debian forky.

  Existing **fcitx** users are encouraged to switch to **fcitx5** following the fcitx upstream migration guide and Debian Wiki page.

5.4 Known severe bugs

Although Debian releases when it's ready, that unfortunately doesn't mean there are no known bugs. As part of the release process all the bugs of severity serious or higher are actively tracked by the Release Team, so an overview of those bugs that were tagged to be ignored in the last part of releasing trixie can be found in the Debian Bug Tracking System. The following bugs were affecting trixie at the time of the release and worth mentioning in this document:

Bug number	Package (source or binary)	Description
1032240	**akonadi-backend-mysql**	akonadi server fails to start since it cannot connect to mysql database
1032177	**faketime**	faketime doesn't fake time (on i386)
918984	**src:fuse3**	provide upgrade path fuse -> fuse3 for bookworm
1016903	**g++-12**	tree-vectorize: Wrong code at O2 level (-fno-tree-vectorize is working)
1034752	**src:gluegen2**	embeds non-free headers

CHAPTER

SIX

MORE INFORMATION ON DEBIAN

6.1 Further reading

Beyond these release notes and the installation guide (at https://www.debian.org/releases/trixie/installmanual) further documentation on Debian is available from the Debian Documentation Project (DDP), whose goal is to create high-quality documentation for Debian users and developers, such as the Debian Reference, Debian New Maintainers Guide, the Debian FAQ, and many more. For full details of the existing resources see the Debian Documentation website and the Debian Wiki.

Documentation for individual packages is installed into `/usr/share/doc/package`. This may include copyright information, Debian specific details, and any upstream documentation.

6.2 Getting help

There are many sources of help, advice, and support for Debian users, though these should only be considered after researching the issue in available documentation. This section provides a short introduction to these sources which may be helpful for new Debian users.

6.2.1 Mailing lists

The mailing lists of most interest to Debian users are the debian-user list (English) and other debian-user-language lists (for other languages). For information on these lists and details of how to subscribe see https://lists.debian.org/. Please check the archives for answers to your question prior to posting and also adhere to standard list etiquette.

6.2.2 Internet Relay Chat

Debian has an IRC channel dedicated to support and aid for Debian users, located on the OFTC IRC network. To access the channel, point your favorite IRC client at irc.debian.org and join `#debian`.

Please follow the channel guidelines, respecting other users fully. The guidelines are available at the Debian Wiki.

For more information on OFTC please visit the website.

6.3 Reporting bugs

We strive to make Debian a high-quality operating system; however that does not mean that the packages we provide are totally free of bugs. Consistent with Debian's "open development" philosophy and as a service to our users, we provide all the information on reported bugs at our own Bug Tracking System (BTS). The BTS can be browsed at https://bugs.debian.org/.

If you find a bug in the distribution or in packaged software that is part of it, please report it so that it can be properly fixed for future releases. Reporting bugs requires a valid e-mail address. We ask for this so that we can trace bugs and developers can get in contact with submitters should additional information be needed.

You can submit a bug report using the program `reportbug` or manually using e-mail. You can find out more about the Bug Tracking System and how to use it by reading the reference documentation (available at `/usr/share/doc/debian` if you have **doc-debian** installed) or online at the Bug Tracking System.

6.4 Contributing to Debian

You do not need to be an expert to contribute to Debian. By assisting users with problems on the various user support lists you are contributing to the community. Identifying (and also solving) problems related to the development of the distribution by participating on the development lists is also extremely helpful. To maintain Debian's high-quality distribution, submit bugs and help developers track them down and fix them. The tool `how-can-i-help` helps you to find suitable reported bugs to work on. If you have a way with words then you may want to contribute more actively by helping to write documentation or translating existing documentation into your own language.

If you can dedicate more time, you could manage a piece of the Free Software collection within Debian. Especially helpful is if people adopt or maintain items that people have requested for inclusion within Debian. The Work Needing and Prospective Packages database details this information. If you have an interest in specific groups then you may find enjoyment in contributing to some of Debian's subprojects which include ports to particular architectures and Debian Pure Blends for specific user groups, among many others.

In any case, if you are working in the free software community in any way, as a user, programmer, writer, or translator you are already helping the free software effort. Contributing is rewarding and fun, and as well as allowing you to meet new people it gives you that warm fuzzy feeling inside.

CHAPTER

SEVEN

MANAGING YOUR BOOKWORM SYSTEM BEFORE THE UPGRADE

This appendix contains information on how to make sure you can install or upgrade bookworm packages before you upgrade to trixie.

7.1 Upgrading your bookworm system

Basically this is no different from any other upgrade of bookworm you've been doing. The only difference is that you first need to make sure your package list still contains references to bookworm as explained in *Checking your APT source-list files*.

If you upgrade your system using a Debian mirror, it will automatically be upgraded to the latest bookworm point release.

7.2 Checking your APT source-list files

If any of the lines in your APT source-list files (see sources.list(5)) contain references to "stable", this is effectively pointing to trixie already. This might not be what you want if you are not yet ready for the upgrade. If you have already run `apt update`, you can still get back without problems by following the procedure below.

If you have also already installed packages from trixie, there probably is not much point in installing packages from bookworm anymore. In that case you will have to decide for yourself whether you want to continue or not. It is possible to downgrade packages, but that is not covered here.

As root, open the relevant APT source-list file (such as `/etc/apt/sources.list`) with your favorite editor, and check all lines beginning with

- `deb http:`
- `deb https:`
- `deb tor+http:`
- `deb tor+https:`
- `URIs: http:`
- `URIs: https:`
- `URIs: tor+http:`
- `URIs: tor+https:`

for a reference to "stable". If you find any, change "stable" to "bookworm".

If you have any lines starting with `deb file:` or `URIs: file:`, you will have to check for yourself if the location they refer to contains a bookworm or trixie archive.

Important: Do not change any lines that begin with `deb cdrom:` or `URIs: cdrom:`. Doing so would invalidate the line and you would have to run `apt-cdrom` again. Do not be alarmed if a `cdrom:` source line refers to "unstable". Although confusing, this is normal.

If you've made any changes, save the file and execute

```
# apt update
```

to refresh the package list.

7.3 Performing the upgrade to latest bookworm release

To upgrade all packages to the state of the latest point release for bookworm, do

```
# apt full-upgrade
```

7.4 Removing obsolete configuration files

Before upgrading your system to trixie, it is recommended to remove old configuration files (such as `*.dpkg-{new,old}` files under `/etc`) from the system.

CHAPTER

EIGHT

CONTRIBUTORS TO THE RELEASE NOTES

Many people helped with the release notes, including, but not limited to

- Adam D. Barrat (various fixes in 2013),
- Adam Di Carlo (previous releases),
- Andreas Barth aba (previous releases: 2005 - 2007),
- Andrei Popescu (various contributions),
- Anne Bezemer (previous release),
- Bob Hilliard (previous release),
- Charles Plessy (description of GM965 issue),
- Christian Perrier bubulle (Lenny installation),
- Christoph Berg (PostgreSQL-specific issues),
- Daniel Baumann (Debian Live),
- David Prévot taffit (Wheezy release),
- Eddy Petrișor (various contributions),
- Emmanuel Kasper (backports),
- Esko Arajärvi (rework X11 upgrade),
- Frans Pop fjp (previous release Etch),
- Giovanni Rapagnani (innumerable contributions),
- Gordon Farquharson (ARM port issues),
- Hideki Yamane henrich (contributed and contributing since 2006),
- Holger Wansing holgerw (contributed and contributing since 2009),
- Javier Fernández-Sanguino Peña jfs (Etch release, Squeeze release),
- Jens Seidel (German translation, innumerable contributions),
- Jonas Meurer (syslog issues),
- Jonathan Nieder (Squeeze release, Wheezy release),
- Joost van Baal-Ilić joostvb (Wheezy release, Jessie release),
- Josip Rodin (previous releases),
- Julien Cristau jcristau (Squeeze release, Wheezy release),

- Justin B Rye (English fixes),
- LaMont Jones (description of NFS issues),
- Luk Claes (editors motivation manager),
- Martin Michlmayr (ARM port issues),
- Michael Biebl (syslog issues),
- Moritz Mühlenhoff (various contributions),
- Niels Thykier nthykier (Jessie release),
- Noah Meyerhans (innumerable contributions),
- Noritada Kobayashi (Japanese translation (coordination), innumerable contributions),
- Osamu Aoki (various contributions),
- Paul Gevers elbrus (buster release),
- Peter Green (kernel version note),
- Rob Bradford (Etch release),
- Samuel Thibault (description of d-i Braille support),
- Simon Bienlein (description of d-i Braille support),
- Simon Paillard spaillar-guest (innumerable contributions),
- Stefan Fritsch (description of Apache issues),
- Steve Langasek (Etch release),
- Steve McIntyre (Debian CDs),
- Tobias Scherer (description of "proposed-update"),
- victory victory-guest (markup fixes, contributed and contributing since 2006),
- Vincent McIntyre (description of "proposed-update"),
- W. Martin Borgert (editing Lenny release, switch to DocBook XML).

This document has been translated into many languages. Many thanks to all the translators!