Chapter

1 The Debian Project

Contents

Before diving right into the technology, let us have a look at what the Debian Project is, its objectives, its means, and its operations.

1.1. What Is Debian?

CULTURE
Origin of the Debian name

Look no further: Debian is not an acronym. This name is, in reality, a contraction of two first names: that of Ian Murdock, and his girlfriend at the time, Debra. Debra + Ian = Debian.

Debian is a GNU/Linux distribution. We will discuss what a distribution is in further detail in section 1.5, "The Role of Distributions" page 23, but for now, we will simply state that it is a complete operating system, including software and systems for installation and management, all based on the Linux kernel and free software (especially those from the GNU project).

When he created Debian, in 1993, under the leadership of the FSF, Ian Murdock had clear objectives, which he expressed in the *Debian Manifesto*. The free operating system that he sought would have to have two principal features. First, quality: Debian would be developed with the greatest care, to be worthy of the Linux kernel. It would also be a non-commercial distribution, sufficiently credible to compete with major commercial distributions. This double ambition would, in his eyes, only be achieved by opening the Debian development process just like that of Linux and the GNU project. Thus, peer review would continuously improve the product.

CULTURE
GNU, the project of the FSF

The GNU project is a range of free software developed, or sponsored, by the Free Software Foundation (FSF), originated by its iconic leader, Dr. Richard M. Stallman. GNU is a recursive acronym, standing for "GNU is Not Unix".

CULTURE
Richard Stallman

FSF's founder and author of the GPL license, Richard M. Stallman (often referred to by his initials, RMS) is a charismatic leader of the Free Software movement. Due to his uncompromising positions, he is not unanimously admired, but his non-technical contributions to Free Software (in particular, the legal and philosophical) are respected by everybody.

1.1.1. A Multi-Platform Operating System

COMMUNITY
Ian Murdock's journey

Ian Murdock, founder of the Debian project, was its first leader, from 1993 to 1996. After passing the baton to Bruce Perens, Ian took a less public role. He returned to working behind the scenes of the free software community, creating the Progeny company, with the intention of marketing a distribution derived from Debian. This venture was, sadly, a commercial failure, and development was abandoned. The company, after several years of scraping by, simply as a service provider, eventually filed for bankruptcy in April of 2007. Of the various projects initiated by Progeny, only *discover* still remains. It is an automatic hardware detection tool.

Ian Murdock died on 28 December 2015 in San Francisco after a series of worrying tweets where he reported having been assaulted by police. In July 2016 it was announced that his death had been ruled a suicide.

Debian, remaining true to its initial principles, has had so much success that, today, it has reached a tremendous size. Currently there are 10 hardware architectures officially supported and also other kernels like FreeBSD (although the FreeBSD-based ports are not part of the set of officially supported architectures). Furthermore, with more than 28,000 source packages, the available software can meet almost any need that one could have, whether at home or in the enterprise.

The sheer size of the distribution can be inconvenient: it is really unreasonable to distribute 16 DVD-ROMs to install a complete version on a standard PC... This is why Debian is increasingly considered as a "meta-distribution", from which one extracts more specific distributions intended for a particular public: Debian Science for scientific use, Debian Edu for education and pedagogical use in an academic environment, Debian Med for medical applications, Debian Jr. for young children, etc. A more complete list of the subprojects can be found in section 1.3.3.1, "Existing Debian Sub-Projects" page 18, dedicated to that purpose.

These partial views of Debian are organized in a well-defined framework, thus guaranteeing hassle-free compatibility between the various "sub-distributions". All of them follow the general planning for release of new versions. And since they build on the same foundations, they can be easily extended, completed, and personalized with applications available in the Debian repositories.

All the Debian tools operate in this direction: `debian-cd` has for a long time now allowed the creation of a set of CD-ROMs containing only a pre-selected set of packages; `debian-installer` is also a modular installer, easily adapted to special needs. `APT` will install packages from various origins, while guaranteeing the overall consistency of the system.

TOOL

Creating a Debian CD-ROM

`debian-cd` creates ISO images of installation media (CD, DVD, Blu-Ray, etc.) ready for use. Any matter regarding this software is discussed (in English) on the debian-cd@lists.debian.org mailing list. The team is led by Steve McIntyre who is handling official Debian ISO builds.

BACK TO BASICS

To each computer, its architecture

The term "architecture" indicates a type of computer (the most known include Mac or PC). Each architecture is differentiated primarily according to its processor, usually incompatible with other processors. These differences in hardware involve varying means of operation, thus requiring that software be compiled specifically for each architecture.

Most software available in Debian is written in portable programming languages: the same source code can be compiled for various architectures. In effect, an executable binary, always compiled for a specific architecture, will not usually function on any of the other architectures.

Remember that each program is created by writing source code; this source code is a text file composed of instructions in a given programming language. Before you can use the software, it is necessary to compile the source code, which means transforming the code into a binary (a series of machine instructions executable by the processor). Each programming language has a specific compiler to execute this operation (for example, `gcc` for the C programming language).

TOOL

Installer

`debian-installer` is the name of the Debian installation program. Its modular design allows it to be used in a broad range of installation scenarios. The development work is coordinated on the debian-boot@lists.debian.org mailing list under the direction of Cyril Brulebois.

1.1.2. The Quality of Free Software

Debian follows all of the principles of Free Software, and its new versions are not released until they are ready. Developers do not work upon a set schedule and don't have to rush to meet an arbitrary deadline. People frequently complain of the long time between Debian's stable releases, but this caution ensures that Debian's legendary reliability is met: long months of testing are indeed necessary for the full distribution to receive the "stable" label.

Debian will not compromise on quality: all known critical bugs on key packages are resolved in any new version, even if this requires the initially forecast release date to be pushed back. Optional packages whose critical bugs are not fixed, and thus do not meet the quality requirements, are simply dropped from the stable release.

1.1.3. The Legal Framework: A Non-Profit Organization

Legally speaking, Debian is a project managed by an American not-for-profit, volunteer association. The project has around a thousand *Debian developers*, but brings together a far greater number of contributors (translators, bug reporters, artists, casual developers, etc.).

To carry its mission to fruition, Debian has a large infrastructure, with many servers connected across the Internet, offered and hosted by many sponsors.

COMMUNITY

Behind Debian, the SPI association, and local branches

Debian doesn't own any server in its own name, since it is only a project within the *Software in the Public Interest* (SPI) association, which manages the hardware and financial aspects (donations, purchase of hardware, etc.). Although it was initially created specifically for the Debian project, this association now hosts other free software projects, especially the PostgreSQL database, Freedesktop.org (project for standardization of various parts of modern graphical desktop environments, such as GNOME and KDE Plasma), and the LibreOffice office suite.

➨ `https://www.spi-inc.org/`

In addition to SPI, various local associations collaborate closely with Debian in order to generate funds for Debian, without centralizing everything in the USA: they are known as "Trusted Organizations" in the Debian jargon. This setup avoids prohibitive international transfer costs, and fits well with the decentralized nature of the project.

Do not hesitate to join your local association and support the project!

➨ `https://wiki.debian.org/Teams/Auditor/Organizations`

➨ `https://france.debian.net/`

➨ `https://debian.ch/`

1.2. The Foundation Documents

A few years after its initial launch, Debian formalized the principles that it should follow as a free software project. This deliberately activist decision allows orderly and peaceful growth by ensuring that all members progress in the same direction. To become a Debian developer, any candidate must confirm and prove their support and adherence to the principles established in the project's Foundation Documents.

The development process is constantly debated, but these Foundation Documents are widely and consensually supported, thus rarely change. The Debian constitution also offers other guarantees for their stability: a three-quarter qualified majority is required to approve any amendment.

1.2.1. The Commitment towards Users

The project also has a "social contract". What place does such a text have in a project only intended for the development of an operating system? It is quite simple: Debian works for its users, and thus, by extension, for society. This contract summarizes the commitments that the project undertakes. Let us study them in greater detail:

1. Debian will remain 100% free.

   This is Rule No. 1. Debian is and will remain composed entirely and exclusively of free software. Additionally, all software development within the Debian project, itself, will be free.

   PERSPECTIVE
   Beyond software

   The first version of the Debian Social Contract said "Debian Will Remain 100% Free *Software*". The disappearance of this last word (with the ratification of Version 1.1 of the contract in April of 2004) indicates the will to achieve freedom, not only in software, but also in the documentation and any other element that Debian wishes to provide within its operating system.

   This change, which was only intended as editorial, has, in reality, had numerous consequences, especially with the removal of some problematic documentation. Furthermore, the increasing use of firmware in drivers poses problems: many are non-free, yet they are necessary for proper operation of the corresponding hardware.

2. We will give back to the free software community.

   Any improvement contributed by the Debian project to a work integrated in the distribution is sent back to the author of the work (called "upstream"). In general, Debian will cooperate with the community rather than work in isolation.

   COMMUNITY
   Upstream author, or Debian developer?

   The term "upstream author" means the author(s)/developer(s) of a work, those who write and develop it. On the other hand, a "Debian developer" uses an existing work to make it into a Debian package (the term "Debian maintainer" is better suited).

   In practice, there can be overlaps between both roles: the Debian maintainer may write a patch, which benefits all users of the work. In general,

Debian encourages those in charge of a package in Debian to get involved in "upstream" development as well (they become, then, contributors, without being confined to the role of simple users of a program).

3. We will not hide problems.

   Debian is not perfect, and, there will be new problems to fix every day. Debian will keep its entire bug report database open for public view at all times. Reports that people file on-line will promptly become visible to others.

4. Our priorities are our users and free software.

   This commitment is more difficult to define. Debian imposes, thus, a bias when a decision must be made, and will discard an easy solution for the developers that will jeopardize the user experience, opting for a more elegant solution, even if it is more difficult to implement. This means to take into account, as a priority, the interests of the users and free software.

5. Works that do not meet our free software standards.

   Debian accepts and understands that users may want to use some non-free programs. That is why the project allows usage of parts of its infrastructure to distribute Debian packages of non-free software that can safely be redistributed.

COMMUNITY

For or against the non-free section?

The commitment to maintain a structure to accommodate non-free software (i.e. the "non-free" section, see the sidebar "The `main`, `contrib` and `non-free` archives" page 109) is frequently a subject of debate within the Debian community.

Detractors argue that it turns people away from free software equivalents, and contradicts the principle of serving only the free software cause. Supporters flatly state that most of the non-free packages are "nearly free", and held back by only one or two annoying restrictions (the most common being the prohibition against commercial usage of the software). By distributing these works in the non-free branch, we indirectly explain to the author that their creation would be better known and more widely used if they could be included in the main section. They are, thus, politely invited to alter their license to serve this purpose.

After a first and unfruitful attempt in 2004, the complete removal of the non-free section is unlikely to return to the agenda, especially since it contains many useful documents that were moved simply because they did not meet the new requirements for the main section. This is especially the case for certain software documentation files issued by the GNU project (in particular, Emacs and Make).

The continued existence of the non-free section is a source of occasional friction with the Free Software Foundation, and is the main reason it refuses to officially recommend Debian as an operating system.

1.2.2. The Debian Free Software Guidelines

This reference document defines which software is "free enough" to be included in Debian. If a program's license is in accordance with these principles, it can be included in the main section;

on the contrary, and provided that free distribution is permitted, it may be found in the non-free section. The non-free section is not officially part of Debian; it is an added service provided to users.

More than a selection criteria for Debian, this text has become an authority on the subject of free software, and has served as the basis for the "Open Source Definition". Historically, it is therefore one of the first formal definitions of the concept of "free software".

The GNU General Public License, the BSD License, and the Artistic License are examples of traditional free licenses that follow the 9 points mentioned in this text. Below you will find the text as it is published on the Debian website.

➡ https://www.debian.org/social_contract#guidelines

1. **Free redistribution.** The license of a Debian component may not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license may not require a royalty or other fee for such sale.

2. **Source code.** The program must include source code, and must allow distribution in source code as well as compiled form.

3. **Derived works.** The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. **Integrity of the author's source code.** The license may restrict source code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software (*This is a compromise. The Debian group encourages all authors not to restrict any files, source or binary, from being modified*).

5. **No discrimination against persons or groups.** The license must not discriminate against any person or group of persons.

6. **No discrimination against fields of endeavor.** The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. **Distribution of license.** The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. **License must not be specific to Debian.** The rights attached to the program must not depend on the program being part of a Debian system. If the program is extracted from Debian and used or distributed without Debian but otherwise within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the Debian system.

9. **License must not contaminate other software.** The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be free software.

BACK TO BASICS

Copyleft

Copyleft is a principle that consists in using copyrights to guarantee the freedom of a work and its derivatives, rather than restrict the rights of uses, as is the case with proprietary software. It is, also, a play of words on the term "copyright". Richard Stallman discovered the idea when a friend of his, fond of puns, wrote on an envelope addressed to him: "copyleft: all rights reversed". Copyleft imposes preservation of all initial liberties upon distribution of an original or modified version of a work (usually a program). It is, thus, not possible to distribute a program as proprietary software if it is derived from code from a copyleft released program.

The most well-known family of copyleft licenses is, of course, the GNU General Public License (GPL) and its derivatives, the GNU Lesser General Public License (LGPL), and the GNU Free Documentation License (GFDL). Sadly, the copyleft licenses are generally incompatible with each other. Consequently, it is best to use only one of them.

10. **Example licenses** The "GPL", "BSD", and "Artistic" licenses are examples of licenses that we consider "free".

BACK TO BASICS

Free licenses

The GNU GPL, the BSD license, and the Artistic License all comply with the Debian Free Software Guidelines, even though they are very different.

The GNU GPL, used and promoted by the FSF (Free Software Foundation), is the most common. Its main feature is that it also applies to any derived work that is redistributed: a program incorporating or using GPL code can only be distributed according to its terms. It prohibits, thus, any reuse in a proprietary application. This poses serious problems for the reuse of GPL code in free software incompatible with this license. As such, it is sometimes impossible to link a program published under another free software license with a library distributed under the GPL. On the other hand, this license is very solid in American law: FSF lawyers have participated in the drafting thereof, and have often forced violators to reach an amicable agreement with the FSF without going to court.

➨ `https://www.gnu.org/copyleft/gpl.html`

The BSD license is the least restrictive: everything is permitted, including use of modified BSD code in a proprietary application.

➨ `https://www.opensource.org/licenses/bsd-license.php`

Finally, the Artistic License reaches a compromise between these two others: integration of code in a proprietary application is permitted, but any modification must be published.

➨ `https://www.opensource.org/licenses/artistic-license-2.0.php`

The complete text of these licenses is available in `/usr/share/common-licenses/` on any Debian system (in case of BSD the newer 3-Clause License).

COMMUNITY

Bruce Perens, a controversial leader

Bruce Perens was the second leader of the Debian project, just after Ian Murdock. He was very controversial in his dynamic and authoritarian methods. He, nevertheless, remains an important contributor to Debian, to whom Debian is especially

indebted for the editing of the famous "Debian Free Software Guidelines" (DFSG), an original idea of Ean Schuessler. Subsequently, Bruce would derive from it the famous "Open Source Definition", removing all references to Debian from it.

➨ https://opensource.org/

His departure from the project was quite emotional, but Bruce has remained strongly attached to Debian, since he continues to promote this distribution in political and economic spheres. He still sporadically appears on the e-mail lists to give his advice and present his latest initiatives in favor of Debian.

Last anecdotal point, it was Bruce who was responsible for inspiring the different "codenames" for Debian versions (1.1 — *Rex*, 1.2 — *Buzz*, 1.3 — *Bo*, 2.0 — *Hamm*, 2.1 — *Slink*, 2.2 — *Potato*, 3.0 — *Woody*, 3.1 — *Sarge*, 4.0 — *Etch*, 5.0 — *Lenny*, 6.0 — *Squeeze*, 7 — *Wheezy*, 8 — *Jessie*, 9 — *Stretch*, 10 — *Buster*, 11 (not released yet) — *Bullseye*, 12 (not released yet) — *Bookworm*, *Unstable* — *Sid*). They are taken from the names of characters in the Toy Story movie. This animated film entirely composed of computer graphics was produced by Pixar Studios, with whom Bruce was employed at the time that he led the Debian project. The name "Sid" holds particular status, since it will eternally be associated with the *Unstable* branch. In the film, this character was the neighbor's child who always broke toys — so beware of getting too close to *Unstable*. Otherwise, *Sid* is also an acronym for "Still In Development".

1.3. The Inner Workings of the Debian Project

The abundant end results produced by the Debian project derive simultaneously from the work on the infrastructure performed by experienced Debian developers, from the individual or collective work of developers on Debian packages, and from user feedback.

1.3.1. The Debian Developers

Debian developers have various responsibilities, and as official project members, they have great influence on the direction the project takes. A Debian developer is generally responsible for at least one package, but according to their available time and desire, they are free to become involved in numerous teams, thus acquiring more responsibilities within the project.

➨ https://www.debian.org/devel/people

➨ https://www.debian.org/intro/organization

➨ https://wiki.debian.org/Teams

TOOL

Developer's database

Debian has a database including all developers registered with the project, and their relevant information (address, telephone, geographical coordinates such as longitude and latitude, etc.). Some of the information (first and last name, country, username within the project, IRC username, GnuPG key, etc.) is public and available on the Web.

➨ https://db.debian.org/

The geographical coordinates allow the creation of a map locating all of the developers around the globe. Debian is truly an international project: its developers can be found on all continents, although the majority are in "Western countries".

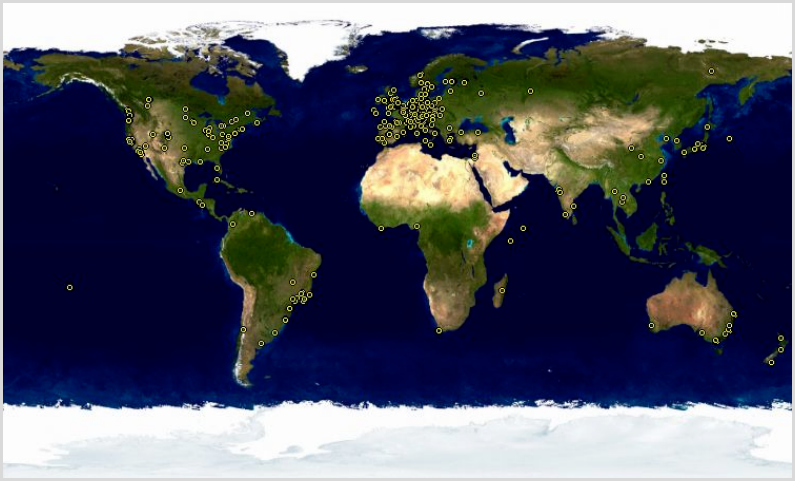

Figure 1.1 *World-wide distribution of Debian developers*

Package maintenance is a relatively regimented activity, very documented or even regulated. It must, in effect, comply with all the standards established by the *Debian Policy*. Fortunately, there are many tools that facilitate the maintainer's work. The developer can, thus, focus on the specifics of their package and on more complex tasks, such as squashing bugs.

➨ `https://www.debian.org/doc/debian-policy/`

BACK TO BASICS

Package maintenance, the developer's work

Maintaining a package entails, first, "packaging" a program. Specifically, this means to define the means of installation so that, once installed, this program will operate and comply with the rules which the Debian project sets for itself. The result of this operation is saved in a `.deb` file. Effective installation of the program will then require nothing more than extraction of this compressed archive and execution of some pre-installation or post-installation scripts contained therein.

After this initial phase, the maintenance cycle truly begins: preparing updates to follow the latest version of the Debian Policy, fixing bugs reported by users, and including new "upstream" versions of the program which naturally continues to develop simultaneously. For instance, at the time of the initial packaging, the program was at version 1.2.3. After some months of development, the original authors release a new stable version, numbered 1.4.0. At this point, the Debian maintainer should update the package, so that users can benefit from its latest stable version.

The Policy, an essential element of the Debian Project, establishes the norms ensuring both the quality of the packages and perfect interoperability of the distribution. Thanks to this Policy, Debian remains consistent despite its gigantic size. This Policy is not fixed in stone, but continuously evolves thanks to proposals formulated on the debian-policy@lists.debian.org mailing

list. Amendments that are agreed upon by all interested parties are accepted and applied to the text by a small group of maintainers who have no editorial responsibility (they only include the modifications agreed upon by the Debian developers that are members of the above-mentioned list). You can read current amendment proposals on the bug tracking system:

➨ https://bugs.debian.org/debian-policy

COMMUNITY

Policy editorial process

Anyone can propose an amendment to the Debian Policy just by submitting a bug report with a severity level of "wishlist" against the *debian-policy* package. The process that then starts is documented in https://www.debian.org/doc/debian-policy/ap-process.html: if it is acknowledged that the problem revealed must be resolved by creating a new rule in the Debian Policy, a discussion begins on the debian-policy@lists.debian.org mailing list until consensus is reached and a proposal issued. Someone then drafts a desired amendment and submits it for approval (in the form of a patch to review). As soon as two other developers approve the fact that the proposed amendment reflects the consensus reached in the previous discussion (they "second" it), the proposal can be included in the official document by one of the *debian-policy* package maintainers. If the process fails at one of these steps, the maintainers close the bug, classifying the proposal as rejected.

DEBIAN POLICY

The documentation

Documentation for each package is stored in `/usr/share/doc/package/`. This directory often contains a `README.Debian` file describing the Debian specific adjustments made by the package maintainer. It is, thus, wise to read this file prior to any configuration, in order to benefit from their experience. We also find a `changelog.Debian.gz` file describing the changes made from one version to the next by the Debian maintainer. This is not to be confused with the `changelog.gz` file (or equivalent), which describes the changes made by the upstream developers. The `copyright` file includes information about the authors and the license covering the software. Finally, we may also find a file named `NEWS.Debian.gz`, which allows the Debian developer to communicate important information regarding updates; if *apt-listchanges* is installed, then these messages are automatically displayed. All other files are specific to the software in question. We especially would like to point out the `examples` sub-directory, which frequently contains examples of configuration files.

The Policy provides considerable cover of the technical aspects of packaging. The size of the project also raises organizational problems; these are dealt with by the Debian Constitution, which establishes a structure and means for decision making. In other words, a formal governance system.

This constitution defines a certain number of roles and positions, plus responsibilities and authorities for each. It is particularly worth noting that Debian developers always have ultimate decision making authority by a vote of general resolution, wherein a qualified majority of three quarters (75%) of votes is required for significant alterations to be made (such as those with an impact on the Foundation Documents). However, developers annually elect a "leader" to represent them in meetings, and ensure internal coordination between varying teams. This election is always a period of intense discussions. This leader's role is not formally defined by any document: candidates for this post usually propose their own definition of the position. In practice,

the leader's roles include serving as a representative to the media, coordinating between "internal" teams, and providing overall guidance to the project, within which the developers can relate: the views of the DPL are implicitly approved by the majority of project members.

Specifically, the leader has real authority; their vote resolves tie votes; they can make any decision which is not already under the authority of someone else and can delegate part of their responsibilities.

Since its inception, the project has been successively led by Ian Murdock, Bruce Perens, Ian Jackson, Wichert Akkerman, Ben Collins, Bdale Garbee, Martin Michlmayr, Branden Robinson, Anthony Towns, Sam Hocevar, Steve McIntyre, Stefano Zacchiroli, Lucas Nussbaum, Mehdi Dogguy, Chris Lamb and Sam Hartman.

The constitution also defines a "technical committee". This committee's essential role is to decide on technical matters when the developers involved have not reached an agreement between themselves. Otherwise, this committee plays an advisory role for any developer who fails to make a decision for which they are responsible. It is important to note that they only get involved when invited to do so by one of the parties in question.

Finally, the constitution defines the position of "project secretary", who is in charge of the organization of votes related to the various elections and general resolutions.

The "general resolution" procedure is fully detailed in the constitution, from the initial discussion period to the final counting of votes. The most interesting aspect of that process is that when it comes to an actual vote, developers have to rank the different ballot options between them and the winner is selected with a Condorcet method[1] (more specifically, the Schulze method). For further details see:

➨ https://www.debian.org/devel/constitution

CULTURE

Flamewar, the discussion that catches fire

A "flamewar" is an exceedingly impassioned debate, which frequently ends up with people attacking each other once all reasonable argumentation has been exhausted on both sides. Certain themes are more frequently subject to polemics than others (the choice of text editor, "do you prefer `vi` or `emacs`?", is an old favorite). The matters often provoke very rapid e-mail exchanges due to the sheer number of people with an opinion on the matter (everyone) and the very personal nature of such questions.

Nothing particularly useful generally comes from such discussions; the general recommendation is to stay out of such debates, and maybe rapidly skim through their content, since reading them in full would be too time-consuming.

Even if this constitution establishes a semblance of democracy, the daily reality is quite different: Debian naturally follows the free software rules of the do-ocracy: the one who does things gets to decide how to do them. A lot of time can be wasted debating the respective merits of various ways to approach a problem; the chosen solution will be the first one that is both functional and satisfying... which will come out of the time that a competent person did put into it.

[1] https://en.wikipedia.org/wiki/Condorcet_method

This is the only way to earn one's stripes: do something useful and show that one has worked well. Many Debian "administrative" teams operate by co-optation, preferring volunteers who have already effectively contributed and proved their competence. The public nature of the work of those teams makes it possible for new contributors to observe and start helping without any special privilege. This is why Debian is often described as a "meritocracy".

CULTURE

Meritocracy, the reign of knowledge

Meritocracy is a form of government in which authority is exercised by those with the greatest merit. For Debian, merit is a measure of competence, which is, itself, assessed by observation of past actions by one or more others within the project (Stefano Zacchiroli, a former project leader, speaks of "do-ocracy", meaning "power to those who get things done"). Their simple existence proves a certain level of competence; their achievements generally being free software, with available source code, which can easily be reviewed by peers to assess their quality.

This effective operational method guarantees the quality of contributors in the "key" Debian teams. This method is by no means perfect and occasionally there are those who do not accept this way of operating. The selection of developers accepted in the teams may appear a bit arbitrary, or even unfair. Furthermore, not everybody has the same definition of the service expected from these teams. For some, it is unacceptable to have to wait eight days for inclusion of a new Debian package, while others will wait patiently for three weeks without a problem. As such, there are regular complaints from the disgruntled about the "quality of service" from some teams.

COMMUNITY

Integration of new maintainers

The team in charge of admitting new developers is the most regularly criticized. One must acknowledge that, throughout the years, the Debian project has become more and more demanding of the developers that it will accept. Some people may see some injustice in that, but we must confess that what were only little challenges at the beginning have become much greater in a community of over 1,000 people, when it comes to ensuring the quality and integrity of everything that Debian produces for its users.

Furthermore, the acceptance procedure is concluded by review of the candidacy by a small team, the Debian Account Managers. These managers are, thus, particularly exposed to criticism, since they have final say in the inclusion or rejection of a volunteer within the Debian developers community. In practice, sometimes they must delay the acceptance of a person until they have learned more about the operations of the project. One can, of course, contribute to Debian before being accepted as an official developer, by being sponsored by current developers.

1.3.2. The Active Role of Users

One might wonder if it is relevant to mention the users among those who work within the Debian project, but the answer is a definite yes: they play a critical role in the project. Far from being "passive", some users run development versions of Debian and regularly file bug reports to indicate problems. Others go even further and submit ideas for improvements, by filing a bug report with a severity level of "wishlist", or even submit corrections to the source code, called "patches" (see section 1.3.2.3, "Sending fixes" page 15).

The fundamental tool for submitting bugs in Debian is the Debian Bug Tracking System (Debian BTS), which is used by large parts of the project. The public part (the web interface) allows users to view all bugs reported, with the option to display a sorted list of bugs selected according to various criteria, such as: affected package, severity, status, address of the reporter, address of the maintainer in charge of it, tag, etc. It is also possible to browse the complete historical listing of all discussions regarding each of the bugs.

Below the surface, the Debian BTS is e-mail based: all information that it stores comes from messages sent by the various persons involved. Any e-mail sent to 12345@bugs.debian.org will, thus, be assigned to the history for bug number 12345. Authorized persons may "close" a bug by writing a message describing the reasons for the decision to close to 12345-done@bugs.debian.org (a bug is closed when the indicated problem is resolved or no longer relevant). A new bug is reported by sending an e-mail to submit@bugs.debian.org according to a specific format which identifies the package in question. The address control@bugs.debian.org allows editing of all the "meta-information" related to a bug.

The Debian BTS has other functional features, as well, such as the use of tags for labeling bugs. For more information, see

➨ `https://www.debian.org/Bugs/`

VOCABULARY

Severity of a bug

The severity of a bug formally assigns a degree of gravity to the reported problem. Effectively, not all bugs have the same importance; for instance, a typo in a manual page is not comparable to a security vulnerability in server software.

Debian uses an extended scale to describe the severity of a bug. Each level is defined precisely in order to facilitate the selection thereof.

➨ `https://www.debian.org/Bugs/Developer#severities`

Users can also use the command line to send bug reports on a Debian package with the `reportbug` tool. It helps making sure the bug in question hasn't already been filed, thus preventing redundancy in the system. It reminds the user of the definitions of the severity levels, for the report to be as accurate as possible (the developer can always fine-tune these parameters later, if needed). It helps writing a complete bug report without the user needing to know the precise syntax, by writing it and allowing the user to edit it. This report will then be sent via an e-mail server (by default, a remote one run by Debian, but `reportbug` can also use a local server).

This tool first targets the development versions, which is where the bugs will be fixed. Effectively, changes are not welcome in a stable version of Debian, with very few exceptions for security updates or other important updates (if, for example, a package is not working at all). A correction of a minor bug in a Debian package must, thus, wait for the next stable version.

Translation and documentation

Additionally, numerous satisfied users of the service offered by Debian like to make a contribution of their own to the project. As not everyone has appropriate levels of expertise in programming, they may choose to assist with the translation and review of documentation. There are language-specific mailing lists to coordinate this work.

➡ https://lists.debian.org/i18n.html

➡ https://www.debian.org/international/

BACK TO BASICS

What are i18n and l10n?

"i18n" and "l10n" are the abbreviations for the words "internationalization" and "localization", respectively, preserving the initial and last letter of each word, and the number of letters in the middle.

To "internationalize" a program consists of modifying it so that it can be translated (localized). This involves partially rewriting a program initially written to work in one language in order to be able to open it to all languages.

To "localize" a program consists of translating the original messages (frequently in English) to another language. For this, it must have already been internationalized.

In summary, internationalization prepares the software for translation, which is then executed by localization.

Sending fixes

More advanced users might be able to provide a fix to a program by sending a patch.

A patch is a file describing changes to be made to one or more reference files. Specifically, it will contain a list of lines to be removed or added to the code, as well as (sometimes) lines taken from the reference text, replacing the modifications in context (they allow identification of the placement of the changes if the line numbers have changed).

The tool used for applying the modifications given in such a file is simply called `patch`. The tool that creates it is called `diff`, and is used as follows:

```
$ diff -u file.old file.new >file.patch
```

The `file.patch` file contains the instructions for changing the content of `file.old` into `file.new`. We can send it to someone, who can then use it to recreate `file.new` from the two others, like this:

```
$ patch -p0 file.old <file.patch
```

The file, `file.old`, is now identical to `file.new`.

In practice, most software is maintained in Git repositories and contributors are thus more likely to use `git` to retrieve the source code and propose changes. `git diff` will generate a file in the same format as what `diff -u` would do and `git apply` can do the same as `patch`.

CULTURE

Git

Git is a tool for collaborative work on multiple files, while maintaining a history of modifications. The files in question are generally text files, such as a program's source code. If several people work together on the same file, `git` can only merge the alterations made if they were made to different portions of the file. Otherwise, these "conflicts" must be resolved by hand.

Git is a distributed system where each user has a repository with the complete history of changes. Central repositories are used to download the project (`git clone`) and to share the work done with others (`git push`). The repository can contain multiple versions of the files but only one version can be worked on at a given time: it is called the working copy (it can be changed to point to another version with `git checkout`). Git can show you the modifications made to the working copy (`git diff`), can stage changes for inclusion (`git add`)), and can create a new entry in the versions history (`git commit`). It can also update the working copy to include modifications made in parallel by other users (`git pull`), and can record a particular configuration in the history in order to be able to easily extract it later on (`git tag`).

Git makes it easy to handle multiple concurrent versions of a project in development without them interfering with each other. These versions are called *branches*. This metaphor of a tree is fairly accurate, since a program is initially developed on a common trunk. When a milestone has been reached (such as version 1.0), development continues on two branches: the development branch prepares the next major release, and the maintenance branch manages updates and fixes for version 1.0.

Git is, nowadays, the most popular version control system but it is not the only one. Historically, CVS (Concurrent Versions System) was the first widely used tool but its numerous limitations contributed to the appearance of more modern free alternatives. These include, especially, `subversion` (`svn`), `git`, `bazaar` (`bzr`), and `mercurial` (`hg`).

➨ `https://www.nongnu.org/cvs/`

➨ `https://subversion.apache.org/`

➨ `https://git-scm.com/`

➨ `https://bazaar.canonical.com/`

➨ `http://mercurial.selenic.com/`

It is beyond the scope of this book to provide a detailed explanation about Git, for that you can refer to the *Pro Git* book.

➨ `https://git-scm.com/book/`

While the output of `git diff` is a file that can be shared with other developers, there are usually better ways to submit changes. If the developers prefer to get patches by email, they usually want patches generated with `git format-patch` so that they can be directly integrated in the repository with `git am`. This preserves commits meta-information and makes it possible to share multiple commits at once.

This email-based workflow is still popular but it tends to be replaced by the usage of *merge requests* (or *pull requests*) whenever the software is hosted in a platform like Github or GitLab — and Debian is using GitLab on its salsa.debian.org server. On those systems, once you have created an account, you *fork* the repository, effectively creating a copy of the repository in your own

account, and you can then clone that repository and push your own changes in it. From there, the web interface will suggest you to submit a merge request, notifying the developers of your changes, making it easy for them to review and accept your changes with a single click.

Other ways of contributing

All of these contribution mechanisms are made more efficient by users' behavior. Far from being a collection of isolated persons, users are a true community within which numerous exchanges take place. We especially note the impressive activity on the user discussion mailing list, debian-user@lists.debian.org (chapter 7, "Solving Problems and Finding Relevant Information" page 148 discusses this in greater detail).

➨ `https://lists.debian.org/users.html`

Not only do users help themselves (and others) on technical issues that directly affect them, but they also discuss the best ways to contribute to the Debian project and help it move forward — discussions that frequently result in suggestions for improvements.

TOOL

how-can-i-help

The `how-can-i-help` program lists opportunities for contributing to Debian packages that are installed locally. After each APT invocation, it shows ways to help, by highlighting bugs tagged "newcomer" (which are easy entry-points for new contributors) or orphaned packages that need a new maintainer. The program can be executed directly as well.

Since Debian does not expend funds on any self-promoting marketing campaigns, its users play an essential role in its diffusion, ensuring its fame via word-of-mouth.

This method works quite well, since Debian fans are found at all levels of the free software community: from install parties (workshops where seasoned users assist newcomers to install the system) organized by local LUGs or "Linux User Groups", to association booths at large tech conventions dealing with Linux, etc.

Volunteers make posters, brochures, stickers, and other useful promotional materials for the project, which they make available to everyone, and which Debian provides freely on its website and on its wiki:

➨ `https://www.debian.org/events/material`

1.3.3. Teams and Sub-Projects

Debian has been organized, right from the start, around the concept of source packages, each with its maintainer or group of maintainers. Many work teams have emerged over time, ensuring administration of the infrastructure, management of tasks not specific to any package in particular (quality assurance, Debian Policy, installer, etc.), with the latest series of teams growing up around sub-projects.

To each their own Debian! A sub-project is a group of volunteers interested in adapting Debian to specific needs. Beyond the selection of a sub-group of programs intended for a particular domain (education, medicine, multimedia creation, etc.), sub-projects are also involved in improving existing packages, packaging missing software, adapting the installer, creating specific documentation, and more.

VOCABULARY

Sub-project and derivative distribution

The development process for a derivative distribution consists in starting with a particular version of Debian and making a number of modifications to it. The infrastructure used for this work is completely external to the Debian project. There isn't necessarily a policy for contributing improvements. This difference explains how a derivative distribution may "diverge" from its origins, and why they have to regularly resynchronize with their source in order to benefit from improvements made upstream.

On the other hand, a sub-project can not diverge, since all the work on it consists of directly improving Debian in order to adapt it to a specific goal.

The most known distribution derived from Debian is, without a doubt, Ubuntu, but there are many. See appendix A, "Derivative Distributions" page 469 to learn about their particularities and their positioning in relationship to Debian.

Here is a small selection of current sub-projects:

- Debian Jr., by Ben Armstrong, offering an appealing and easy to use Debian system for children;
- Debian Edu, by Petter Reinholdtsen, focused on the creation of a specialized distribution for the academic world;
- Debian Med, by Andreas Tille, dedicated to the medical field;
- Debian Multimedia which deals with audio and multimedia work;
- Debian GIS which takes care of Geographical Information Systems applications and users;
- Debian Accessibility, improving Debian to match the requirements of people with disabilities;
- Debian Science, finally, working on providing researchers and scientists a better experience using Debian.
- DebiChem, targeted at Chemistry, provides chemical suites and programs.

The number of projects will most likely continue to grow with time and improved perception of the advantages of Debian sub-projects. Fully supported by the existing Debian infrastructure, they can, in effect, focus on work with real added value, without worrying about remaining synchronized with Debian, since they are developed within the project.

Most administrative teams are relatively closed and recruit only by co-optation. The best means to become a part of one is to intelligently assist the current members, demonstrating that you have understood their objectives and methods of operation.

The *ftpmasters* are in charge of the official archive of Debian packages. They maintain the program that receives packages sent by developers and automatically stores them, after some checks, on the reference server (ftp-master.debian.org).

They must also verify the licenses of all new packages, in order to ensure that Debian may distribute them, prior to including them in the corpus of existing packages. When a developer wishes to remove a package, they address this team through the bug tracking system and the *ftp.debian.org* "pseudo-package".

VOCABULARY

The pseudo-package, a monitoring tool

The bug tracking system, initially designed to associate bug reports with a Debian package, has proved very practical to manage other matters: lists of problems to be resolved or tasks to manage without any link to a particular Debian package. The "pseudo-packages" allow, thus, certain teams to use the bug tracking system without associating a real package with their team. Everyone can, thus, report issues that needs to be dealt with. For instance, the BTS has a *ftp.debian.org* entry that is used to report and track problems on the official package archive or simply to request removal of a package. Likewise, the *www.debian.org* pseudo-package refers to errors on the Debian website, and *lists.debian.org* gathers all the problems concerning the mailing lists.

TOOL

GitLab, Git repository hosting and much more

A GitLab instance, known as `salsa.debian.org`, is used by Debian to host the Git packaging repositories but this software offers much more than simple hosting and Debian contributors have been quick to leverage the continuous integration features (running tests, or even building packages, on each push). Debian contributors also benefit from a cleaner contribution workflow thanks the well understood merge request process (similar to GitHub's pull requests).

GitLab replaced FusionForge (which was running on a service known as `alioth.debian.org`) for collaborative package maintenance. This service is administered by Alexander Wirt, Bastian Blank and Jörg Jaspert.

➨ `https://salsa.debian.org/`

➨ `https://wiki.debian.org/Salsa/Doc`

The *Debian System Administrators* (DSA) team (debian-admin@lists.debian.org), as one might expect, is responsible for system administration of the many servers used by the project. They ensure optimal functioning of all base services (DNS, Web, e-mail, shell, etc.), install software requested by Debian developers, and take all precautions in regards to security.

➨ `https://dsa.debian.org`

TOOL

Debian Package Tracker

This is one of Raphaël's creations. The basic idea is, for a given package, to centralize as much information as possible on a single page. Thus, one can quickly check the status of a program, identify tasks to be completed, and offer one's assistance. This is why this page gathers all bug statistics, available versions in each distribution, progress of a package in the *Testing* distribution, the status of translations of descriptions and debconf templates, the possible availability of a new upstream version, notices of noncompliance with the latest version of the Debian Policy, information on the maintainer, and any other information that said maintainer wishes to include.

➨ https://tracker.debian.org/

An e-mail subscription service completes this web interface. It automatically sends the following selected information to the list: bugs and related discussions, availability of a new version on the Debian servers, new translations available for proofreading, etc.

Advanced users can, thus, follow all of this information closely and even contribute to the project, once they have got a good enough understanding of how it works.

Another web interface, known as *Debian Developer's Packages Overview* (DDPO), provides each developer a synopsis of the status of all Debian packages placed under their charge.

➨ https://qa.debian.org/developer.php

These two websites are tools developed and managed by the group responsible for quality assurance within Debian (known as Debian QA).

The *listmasters* administer the e-mail server that manages the mailing lists. They create new lists, handle bounces (delivery failure notices), and maintain spam filters (unsolicited bulk e-mail).

CULTURE

Traffic on the mailing lists: some figures

The mailing lists are, without a doubt, the best testimony to activity on a project, since they keep track of everything that happens. The numbers (from May 2019) regarding our mailing lists speak for themselves: Debian hosts about 315 lists, totaling over 303,000 individual subscriptions. 227,000 e-mails are delivered every day.

Each specific service has its own administration team, generally composed of volunteers who have installed it (and also frequently programmed the corresponding tools themselves). This is the case of the bug tracking system (BTS), the package tracker, salsa.debian.org (GitLab server, see sidebar "GitLab, Git repository hosting and much more" page 19), the services available on qa.debian.org, lintian.debian.org, buildd.debian.org, cdimage.debian.org, etc.

Development Teams, Transversal Teams

Unlike administrative teams, the development teams are rather widely open, even to outside contributors. Even if Debian does not have a vocation to create software, the project needs some specific programs to meet its goals. Of course, developed under a free software license, these tools make use of methods proven elsewhere in the free software world.

Debian has developed little software of its own, but certain programs have assumed a starring role, and their fame has spread beyond the scope of the project. Good examples are `dpkg`, the Debian package management program (it is, in fact, an abbreviation of Debian PacKaGe, and generally pronounced as "dee-package"), and `apt`, a tool to automatically install any Debian package, and its dependencies, guaranteeing the consistency of the system after an upgrade (its name is an acronym for Advanced Package Tool). Their teams are, however, much smaller, since a rather high level of programming skill is required to gain an overall understanding of the operations of these types of programs.

The most important team is probably that for the Debian installation program, `debian-installer`, which has accomplished a work of momentous proportions since its conception in 2001. Numerous contributors were needed, since it is difficult to write a single program able to install Debian on a dozen different architectures. Each one has its own mechanism for booting and its own bootloader. All of this work is coordinated on the debian-boot@lists.debian.org mailing list, under the direction of Cyril Brulebois.

➨ `https://www.debian.org/devel/debian-installer/`

➨ `https://joeyh.name/blog/entry/d-i_retrospective/`

The (very small) `debian-cd` program team has an even more modest objective. Many "small" contributors are responsible for their architecture, since the main developer can not know all the subtleties, nor the exact way to start the installer from the CD-ROM.

Many teams must collaborate with others in the activity of packaging: debian-qa@lists.debian.org tries, for example, to ensure quality at all levels of the Debian project. The debian-policy@lists.debian.org list develops Debian Policy according to proposals from all over the place. The teams in charge of each architecture (debian-*architecture*@lists.debian.org) compile all packages, adapting them to their particular architecture, if needed.

Other teams manage the most important packages in order to ensure maintenance without placing too heavy a load on a single pair of shoulders; this is the case with the C library and debian-glibc@lists.debian.org, the C compiler on the debian-gcc@lists.debian.org list, or Xorg on the debian-x@lists.debian.org (this group is also known as the X Strike Force).

1.4. Follow Debian News

As already mentioned, the Debian project evolves in a very distributed, very organic way. As a consequence, it may be difficult at times to stay in touch with what happens within the project without being overwhelmed with a never-ending flood of notifications.

If you only want the most important news about Debian, you probably should subscribe to the debian-announce@lists.debian.org list. This is a very low-traffic list (around a dozen messages a year), and only gives the most important announcements, such as the availability of a new stable release, the election of a new Project Leader, or the yearly Debian Conference.

➨ `https://lists.debian.org/debian-announce/`

More general (and regular) news about Debian are sent to the debian-news@lists.debian.org list. The traffic on this list is quite reasonable too (usually around a handful of messages a month), and it includes the semi-regular "Debian Project News", which is a compilation of various small bits of information about what happens in the project.

➨ https://lists.debian.org/debian-news/

COMMUNITY

The publicity team

Debian's official communication channels are managed by volunteers of the Debian publicity team. They are delegates of the Debian Project Leader and moderate news and announcements posted there. Many other volunteers contribute to the team, for example, by writing content for "Debian Project News", Debian's official blog (bits.debian.org[2]) or the microblogging service (micronews.debian.org[3]), which supplies social networking sites with microblogging content.

➨ https://wiki.debian.org/Teams/Publicity

For more information about the evolution of Debian and what is happening at some point in time in various teams, there is also the debian-devel-announce@lists.debian.org list. As its name implies, the announcements it carries will probably be more interesting to developers, but it also allows interested parties to keep an eye on what happens in more concrete terms than just when a stable version is released. While debian-announce@lists.debian.org gives news about the user-visible results, debian-devel-announce@lists.debian.org gives news about how these results are produced. As a side note, "d-d-a" (as it is sometimes referred to) is the only list that Debian developers must be subscribed to.

➨ https://lists.debian.org/debian-devel-announce/

Debian's official blog (bits.debian.org[4]) is also a good source of information. It conveys most of the interesting news that are published on the various mailing lists that we already covered and other important news contributed by community members. Since all Debian developers can contribute these news when they think they have something noteworthy to make public, Debian's blog gives a valuable insight while staying rather focused on the project as a whole.

A more informal source of information can also be found on Planet Debian, which aggregates articles posted by Debian contributors on their respective blogs. While the contents do not deal exclusively with Debian development, they provide a view into what is happening in the community and what its members are up to.

➨ https://planet.debian.org/

The project is also well represented on social networks. Debian only has an official presence on Identi.ca (microblogging platform, powered by *pump.io*), but there are some accounts retransmitting the RSS feed from https://micronews.debian.org/ and many Debian contributors who are posting on non-official accounts.

➨ https://identi.ca/debian

[2] https://bits.debian.org
[3] https://micronews.debian.org/
[4] https://bits.debian.org

➨ https://fosstodon.org/@debian

➨ https://twitter.com/debian

➨ https://www.facebook.com/debian

➨ https://www.flickr.com/groups/debian

➨ https://www.linkedin.com/company/debian

1.5. The Role of Distributions

A GNU/Linux distribution has two main objectives: install a free operating system on a computer (either with or without an existing system or systems), and provide a range of software covering all of the users' needs.

1.5.1. The Installer: `debian-installer`

The `debian-installer`, designed to be extremely modular in order to be as generic as possible, targets the first objective. It covers a broad range of installation situations and in general, greatly facilitates the creation of a derivative installer corresponding to a particular case.

This modularity, which also makes it very complex, may be daunting for the developers discovering this tool; but whether used in graphical or text mode, the user's experience is still similar. Great efforts have been made to reduce the number of questions asked at installation time, in particular thanks to the inclusion of automatic hardware detection software.

It is interesting to note that distributions derived from Debian differ greatly on this aspect, and provide a more limited installer (often confined to the i386 or amd64 architectures), but more user-friendly for the uninitiated. On the other hand, they usually refrain from straying too far from package contents in order to benefit as much as possible from the vast range of software offered without causing compatibility problems.

1.5.2. The Software Library

Quantitatively, Debian is undeniably the leader in this respect, with over 28,000 source packages. Qualitatively, Debian's policy and long testing period prior to releasing a new stable version justify its reputation for stability and consistency. As far as availability, everything is available on-line through many mirrors worldwide, with updates pushed out every six hours.

Many retailers sell DVD-ROMs on the Internet at a very low price (often at cost), the "images" for which are freely available for download. There is only one drawback: the low frequency of releases of new stable versions (their development sometimes takes more than two years), which delays the inclusion of new software.

Most new free software programs quickly find their way into the development version which allows them to be installed. If this requires too many updates due to their dependencies, the

program can also be recompiled for the stable version of Debian (see chapter 15, "Creating a Debian Package" page 448 for more information on this topic).

1.6. Lifecycle of a Release

The project will simultaneously have three to six different versions of each program, named *Experimental*, *Unstable*, *Testing*, *Stable*, *Oldstable*, and even *Oldoldstable*. Each one corresponds to a different phase in development. For a good understanding, let us take a look at a program's journey, from its initial packaging to inclusion in a stable version of Debian.

VOCABULARY
Release
The term "release", in the Debian project, indicates a particular version of a distribution (e.g., "unstable release" means "the unstable version"). It also indicates the public announcement of the launch of any new version (stable).

1.6.1. The *Experimental* Status

First let us take a look at the particular case of the *Experimental* distribution: this is a group of Debian packages corresponding to the software currently in development, and not necessarily completed, explaining its name. Not everything passes through this step; some developers add packages here in order to get feedback from more experienced (or braver) users.

Otherwise, this distribution frequently houses important modifications to base packages, whose integration into *Unstable* with serious bugs would have critical repercussions. It is, thus, a completely isolated distribution, its packages never migrate to another version (except by direct, express intervention of the maintainer or the ftpmasters). It is also not self-contained: only a subset of the existing packages are present in *Experimental*, and it generally does not include the base system. This distribution is therefore mostly useful in combination with another, self-contained, distribution such as *Unstable*.

1.6.2. The *Unstable* Status

Let us turn back to the case of a typical package. The maintainer creates an initial package, which they compile for the *Unstable* version and place on the ftp-master.debian.org server. This first event involves inspection and validation from the ftpmasters. The software is then available in the *Unstable* distribution, which is the "cutting edge" distribution chosen by users who are more concerned with having up-to-date packages than worried about serious bugs. They discover the program and then test it.

If they encounter bugs, they report them to the package's maintainer. The maintainer then regularly prepares corrected versions, which they upload to the server.

Every newly updated package is updated on all Debian mirrors around the world within six hours. The users then test the corrections and search for other problems resulting from the

modifications. Several updates may then occur rapidly. During these times, autobuilder robots come into action. Most frequently, the maintainer has only one traditional PC and has compiled their package on the amd64 (or i386) architecture (or they opted for a source-only upload, thus without any precompiled package); the autobuilders take over and automatically compile versions for all the other architectures. Some compilations may fail; the maintainer will then receive a bug report indicating the problem, which is then to be corrected in the next versions. When the bug is discovered by a specialist for the architecture in question, the bug report may come with a patch ready to use.

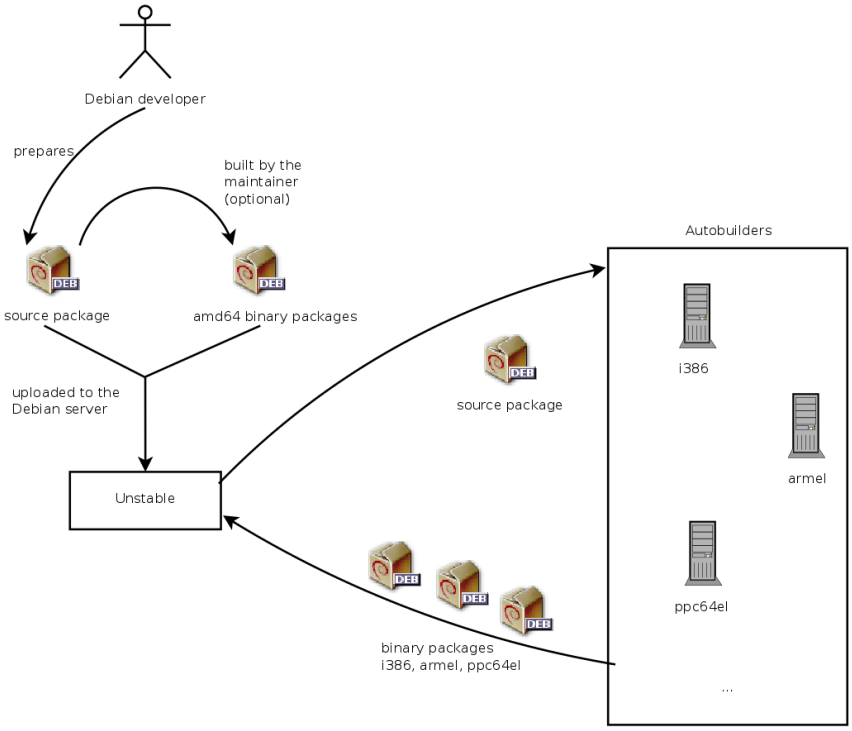

Figure 1.2 *Compilation of a package by the autobuilders*

QUICK LOOK

`buildd`, the Debian package recompiler

buildd is the abbreviation of "build daemon". This program automatically recompiles new versions of Debian packages on the architectures on which it is hosted (cross-compilation is avoided as much as possible).

Thus, to produce binaries for the `arm64` architecture, the project has `arm64` machines available. The *buildd* program runs on them continuously and creates binary packages for `arm64` from source packages sent by Debian developers.

This software is used on all the computers serving as autobuilders for Debian. By extension, the term *buildd* frequently is used to refer to these machines, which are generally reserved solely for this purpose.

1.6.3. Migration to *Testing*

A bit later, the package will have matured; compiled on all the architectures, it will not have undergone recent modifications. It is then a candidate for inclusion in the *Testing* distribution — a group of *Unstable* packages chosen according to some quantifiable criteria. Every day a program automatically selects the packages to include in *Testing*, according to elements guaranteeing a certain level of quality:

1. lack of critical bugs, or, at least fewer than the version currently included in *Testing*;
2. at least 5 days spent in *Unstable*, which is usually sufficient time to find and report any serious problems (successfully passing the package's own test suite, if it has one, reduces that time);
3. successful compilation on all officially supported architectures;
4. dependencies that can be satisfied in *Testing*, or that can at least be moved there together with the package in question;
5. automatic quality tests of the package (*autopkgtest*) — if defined — don't show any regression.

This system is clearly not infallible; critical bugs are regularly found in packages included in *Testing*. Still, it is generally effective, and *Testing* poses far fewer problems than *Unstable*, being for many, a good compromise between stability and novelty.

NOTE

Limitations of *Testing*

While very interesting in principle, *Testing* does have some practical problems: the tangle of cross-dependencies between packages is such that a package can rarely move there completely on its own. With packages all depending upon each other, it is sometimes necessary to migrate a large number of packages simultaneously, which is impossible when some are uploading updates regularly. On the other hand, the script identifying the families of related packages works hard to create them (this would be an NP-complete problem, for which, fortunately, we know some good heuristics). This is why we can manually interact with and guide this script by suggesting groups of packages, or imposing the inclusion of certain packages in a group, even if this temporarily breaks some dependencies. This functionality is accessible to the Release Managers and their assistants.

Recall that an NP-complete problem is of an exponential algorithmic complexity according to the size of the data, here being the length of the code (the number of figures) and the elements involved. The only way to resolve it is frequently to examine all possible configurations, which could require enormous means. A heuristic is an approximate, but satisfying, solution.

COMMUNITY

The Release Manager

Release Manager is an important title, associated with heavy responsibilities. The bearer of this title must, in effect, manage the release of a new, stable version of Debian, and define the process for development of *Testing* until it meets the quality criteria for *Stable*. They also define a tentative schedule (not always followed).

We also have Stable Release Managers, often abbreviated SRM, who manage and select updates for the current stable version of Debian. They systematically include security patches and examine all other proposals for inclusion, on a case by case basis, sent by Debian developers eager to update their package in the stable version.

1.6.4. The Promotion from *Testing* to *Stable*

Let us suppose that our package is now included in *Testing*. As long as it has room for improvement, its maintainer must continue to improve it and restart the process from *Unstable* (but its later inclusion in *Testing* is generally faster: unless it changed significantly, all of its dependencies are already available). When it reaches perfection, the maintainer has completed their work. The next step is the inclusion in the *Stable* distribution, which is, in reality, a simple copy of *Testing* at a moment chosen by the Release Manager. Ideally, this decision is made when the installer is ready, and when no program in *Testing* has any known critical bugs.

Since this moment never truly arrives, in practice, Debian must compromise: remove packages whose maintainer has failed to correct bugs on time, or agree to release a distribution with some bugs in the thousands of programs. The Release Manager will have previously announced a freeze period, during which each update to *Testing* must be approved. The goal here is to prevent any new version (and its new bugs), and to only approve updates fixing bugs.

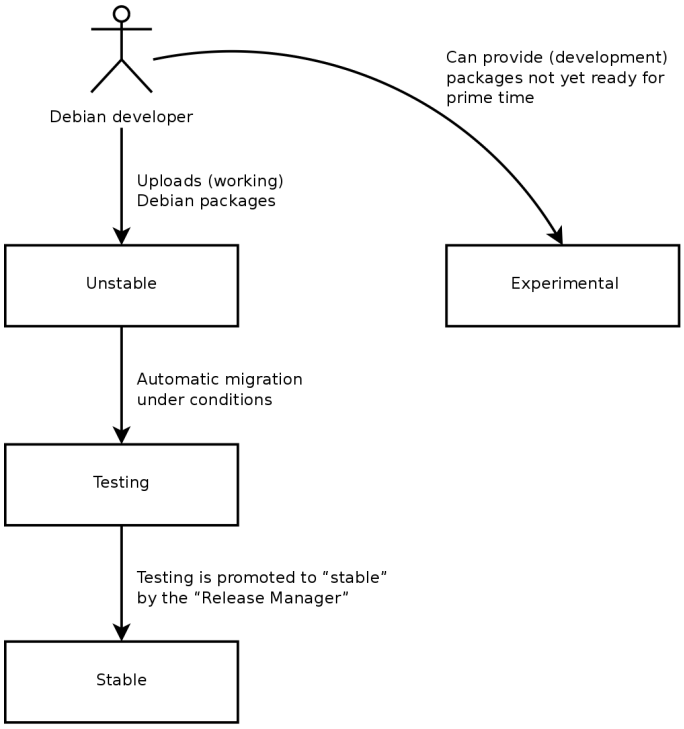

Figure 1.3 *A package's path through the various Debian versions*

VOCABULARY

Freeze: the home straight

During the freeze period, development of the *Testing* distribution is blocked; no more automatic updates are allowed. Only the Release Managers are then authorized to change packages, according to their own criteria. The purpose is to prevent the appearance of new bugs by introducing new versions; only thoroughly examined updates are authorized when they correct significant bugs.

After the release of a new stable version, the Stable Release Managers manage all further development (called "revisions", ex: 7.1, 7.2, 7.3 for version 7). These updates systematically include all security patches. They will also include the most important corrections (the maintainer of a package must prove the gravity of the problem that they wish to correct in order to have their updates included).

At the end of the journey, our hypothetical package is now included in the stable distribution. This journey, not without its difficulties, explains the significant delays separating the Debian Stable releases. This contributes, over all, to its reputation for quality. Furthermore, the majority of users are satisfied using one of the three distributions simultaneously available. The system administrators, concerned above all about the stability of their servers, don't need the latest and greatest version of GNOME; they can choose Debian *Stable*, and they will be satisfied. End users, more interested in the latest versions of GNOME or KDE Plasma than in rock-solid stability, will find Debian *Testing* to be a good compromise between a lack of serious problems and relatively up-to-date software. Finally, developers and more experienced users may blaze the trail, testing all the latest developments in Debian *Unstable* right out of the gate, at the risk of suffering the headaches and bugs inherent in any new version of a program. To each their own Debian!

CULTURE

GNOME and KDE Plasma, graphical desktop environments

GNOME (GNU Network Object Model Environment) and Plasma by KDE (formerly known as K Desktop Environment) are the two most popular graphical desktop environments in the free software world, and will be presented in greater detail in section 13.3, "Graphical Desktops" page 385.

A desktop environment is a set of programs grouped together to allow easy management of the most common operations through a graphical interface. They generally include a file manager, office suite, web browser, e-mail program, multimedia accessories, etc. The most visible difference resides in the choice of the graphical library used: GNOME has chosen GTK+ (free software licensed under the LGPL), and the KDE community has selected Qt (a company-backed project, available nowadays both under the GPL and a commercial license).

➨ https://www.gnome.org/

➨ https://www.kde.org/

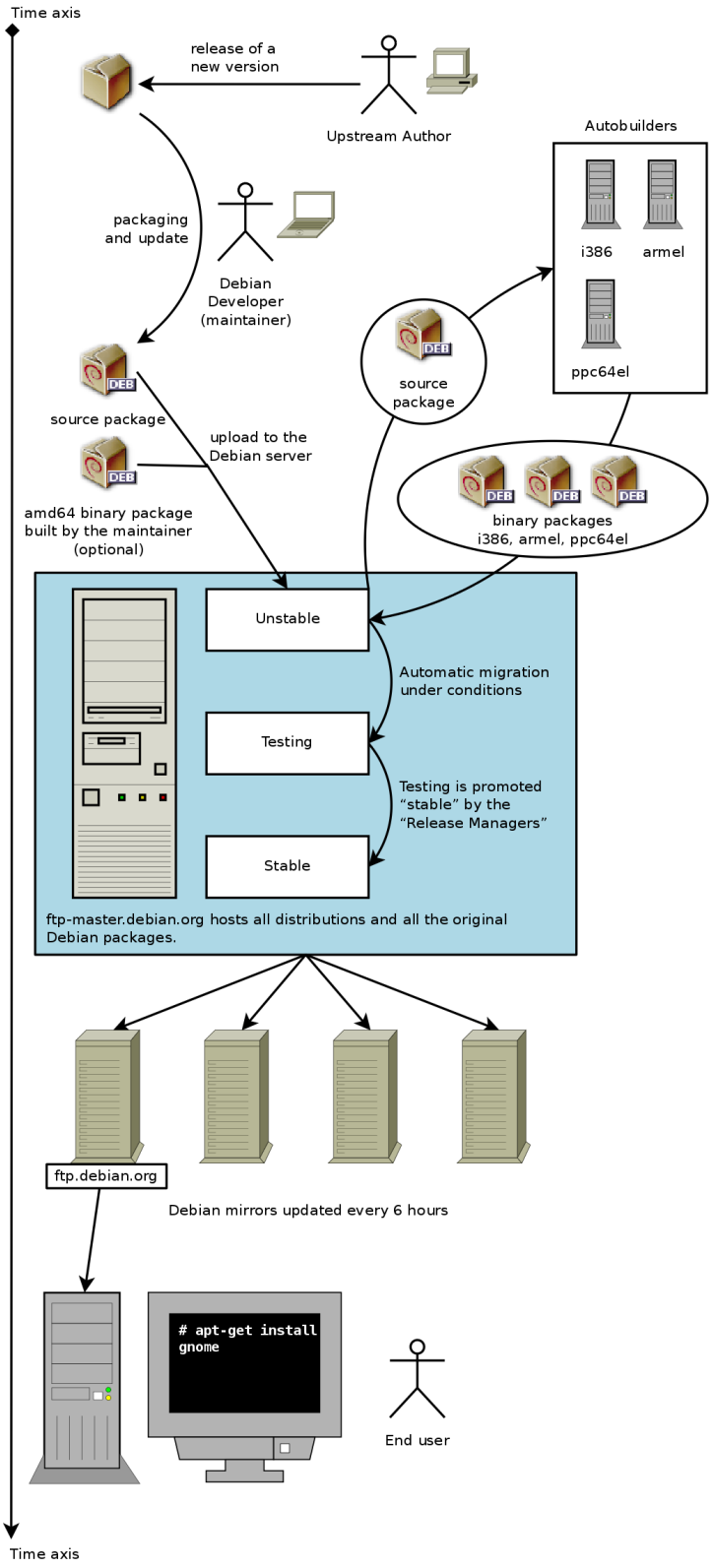

Figure 1.4 *Chronological path of a program packaged by Debian*

1.6.5. The *Oldstable* and *Oldoldstable* Status

Each *Stable* release has an expected lifetime of about 5 years and given that releases tend to happen every 2 years, there can be up to 3 supported releases at a given point of time. When a new stable release happens, the former release becomes *Oldstable* and the one even before becomes *Oldoldstable*.

This Long Term Support (LTS) of Debian releases is a recent initiative: individual contributors and companies joined forces to create the Debian LTS team. Older releases which are no longer supported by the Debian security team fall under the responsibility of this new team.

The Debian security team handles security support in the current *Stable* release and also in the *Oldstable* release (but only for as long as is needed to ensure one year of overlap with the current stable release). This amounts roughly to three years of support for each release. The Debian LTS team handles the last (two) years of security support so that each releases benefits from at least 5 years of support and so that users can upgrade from version N to N+2, for example from Debian 8 "Jessie" to Debian 10 "Buster".

➨ https://wiki.debian.org/LTS

COMMUNITY

Companies sponsoring the LTS effort

Long Term Support is a difficult commitment to make in Debian because volunteers tend to avoid the work that is not very fun. And providing security support for 5 years old software is — for many contributors — a lot less fun than packaging new upstream versions or developing new features.

To bring this project to life, the project counted on the fact that long term support was particularly relevant for companies and that they would be willing to mutualize the cost of this security support.

The project started in June 2014: some organizations allowed their employees to contribute part-time to Debian LTS while others preferred to sponsor the project with money so that Debian contributors get paid to do the work that they would not do for free. Most Debian contributors willing to be paid to work on LTS got together to create a clear sponsorship offer managed by Freexian (Raphaël Hertzog's company):

➨ https://www.freexian.com/services/debian-lts.html

In the Debian LTS team, the volunteers work on packages they care about while the paid contributors prioritize packages used by their sponsors.

The project is always looking for new sponsors: What about your company? Can you let an employee work part-time on long term support? Can you allocate a small budget for security support?

➨ https://wiki.debian.org/LTS/Funding

Keywords

Falcot Corp
SMB
Strong Growth
Master Plan
Migration
Cost Reduction