Martin F. Krafft

The Debian System

Concepts and Techniques

Bibliographic Information of "Die Deutsche Bibliothek"

"Die Deutsche Bibliothek" registers this publication in the "Deutsche Nationalbibliografie"; detailed bibliographic data are available online at **http://dnb.ddb.de**.

Editor: Dr. Markus Wirtz
Cover Design: Fritz Design GmbH, Germany
Printed in Germany on acid-free paper.
ISBN 3-937514-07-4
http://www.opensourcepress.de

of an image not rather consist of all the brush strokes and edits to an image? What if a programme was used to generate the image, which in itself is not free or not part of Debian? What if the image was created in **Adobe Photoshop**?

- Among other things, the GFDL forbids the omission or change of "invariant sections." Such sections can only exist as part of "secondary sections," whose contents must not be concerned with the matter documented in the text. Thus, secondary sections hold copyright and acknowledgements, which are arguably to be propagated to derived works. Is the GFDL then really non-free? Still, invariant sections allow for nasty acts[29], so it limits the freedom. On an unrelated note, removing documentation from the system certainly inconveniences many users and makes Debian unusable (or mostly so) to those without permanent or reasonable Internet access. If the Social Contract promises that Debian will honour the desires of its users, how can this dilemma be resolved?

As you can see, a great number of questions remain unanswered[30]. While Debian hopes to exercise its influence and to cause change, *e.g.* a revised GFDL, or vendors releasing their driver firmware under a free and open licence, it is impossible at this moment to forecast the future of such licence issues. The Debian project promises *etch* to be fully conformant to the Social Contract in any case.

The **debian-legal** mailing list as well as the **#debian-legal** IRC channel **irc.debian.org** are the forums dedicated to discussions that focus on freedom and licencing issues. Please make sure you search the list archives[31] before posting. Also, the Debian Wiki has a page on common licences and their DFSG status[32].

2.4 The Debian community

2.4.1 Organisation of the project

The Debian project is organised according to the structure described in its constitution[33], which establishes the decision-making bodies and the processes for making decisions within the project. Figure 2.2 is a rough approximation of the structure of the Debian project. Many groups, subprojects, and individuals cannot be clearly classified or outlined in a meaningful way. However, the image gives an overview of the most important bodies and their relationships. The list of current occupants of the official positions is available on the Web[34].

[29] Read http://slashdot.org/articles/03/04/20/1357236.shtml for a few examples.
[30] At http://lists.debian.org/debian-vote/2005/03/msg00152.html, you can find a number of challenging licence considerations for different data types.
[31] http://lists.debian.org/debian-legal
[32] http://wiki.debian.net/index.cgi?DFSGLicences
[33] http://www.debian.org/devel/constitution
[34] http://www.debian.org/intro/organization

As shown, the all-encompassing group is the set of users. Within the Debian project, everybody is also a user of the system, which makes for the close relationship between users and developers and also provides most of the motivation for the volunteers running the project. Users can apply to join the project (see chapter 2.5.2), and if approved, they join the developer body. Alternatively, numerous areas exist in which users can contribute to the project without having to go through the application process (see chapter 2.5.1).

The set of developers makes up the major organisational body of Debian. At the time of writing, the Debian project consisted of about 950 developers, each of whom was taking care of one or more parts of the project, be it package management, documentation, internationalisation, organisation, or infrastructure maintenance, to name but a few. Participation in any of these projects is voluntary, generally without an ironclad commitment, which allows people to dedicate their available time to any project they wish.

Possibly one the most important aspects of the Debian community is its international orientation. Even though the project was founded in the United States, Debian developers live all over the world[35], allowing for diversity and political independence, among other traits. As many aspects of the project are security-sensitive, trust among the developers plays an important part (see chapter 2.4.3).

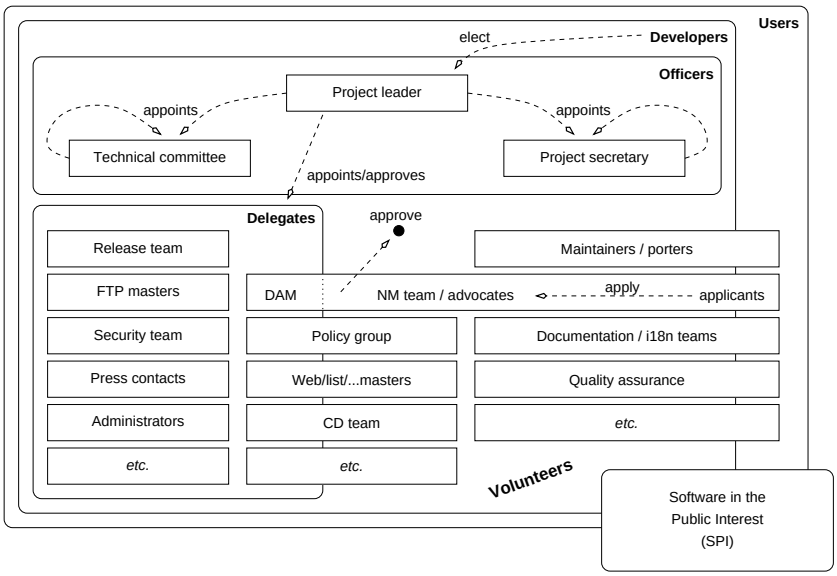

Figure 2.2: A rough sketch of Debian's organisation

[35] http://www.linuks.mine.nu/debian-worldmap

The project officers

The developer body itself is organised in a flat hierarchy. At the top of the hierarchy sits the project leader, who is elected annually by the developers. Every developer has a vote, and every developer can become a candidate. The project leader guides the project and and is responsible for its coordination. The project leader should be a driving force for new directions and can focus developer attention on specific areas of problems. The leader has various special powers which are rarely used. However, the developer collective may cast a majority vote to overrule the project leader's decision, which has never happened.

Upon being elected, the project leader appoints the officers (together with the previous occupants of these positions) and delegates a number of tasks to chosen individuals and teams. Subsequently, the project leader is free to replace officers and delegates as appropriate. The project leader may be replaced by the chairman of the technical committee together with the project secretary, should it be necessary (*e.g.* in case of death or absence without leave).

Assisting the project leader are the members of the technical committee, fronted by the chairman. The technical committee's task is to monitor the technical aspects of the distribution. In exceptional cases, it may require a developer to take a certain course of action over another, if it is deemed to be in the better interest of the project and its users. The project secretary mainly coordinates elections and other votes. Furthermore, the secretary's job is to resolve disputes on the interpretation of the constitution.

The delegates

The delegates receive deputations from the project leader and are given special powers in specific task domains. For instance, the project leader delegates the task of Debian account maintenance to the Debian Account Manager (DAM), and the job of interacting with the media to the press contact. Within their domain, the delegates have the freedom to act according as they judge best. In addition, to prevent concentration of power, the delegates can make certain decisions which the project leader may not make directly. Such decisions are mainly concerned with project membership.

Delegation is not an official or necessarily visible process. In addition, it need not be an explicit process. Many delegate positions are held by people for years, and the project leader simply accepts their status by not choosing replacements. The delegate must have all the necessary competence in the delegated domain to make decisions (or else should not be chosen as a delegate). The project leader cannot override a delegate's decision once it is spoken. Therefore, it is in the leader's best interest to choose someone trustworthy and capable. The project leader may replace delegates given sufficient reasons. However, a single decision made, possibly in disfavour of the leader, is not a sufficient reason.

Notable delegate positions include the release managers, the security team, and the FTP masters. The release managers set the goals for the next ***stable*** release, and coordinate and supervise the release process. This involves scheduling freezes and carving the final release date in stone. A separate ***stable*** release manager then takes over. The security team's job is to cooperate with security teams from other projects to provide security-related fixes to software in the ***stable*** archive as soon as possible. Finally, the FTP masters are in charge of the Debian archive, deciding what software is allowed into the archive and what must be removed.

The developer collective

At the bottom, albeit all-encompassing, resides the remainder of developers, each acting within their own self-assigned niche of the project. Although it is preferred for a developer to maintain a package (a common but not absolutely necessary requirement for becoming a Debian developer; see chapter 2.5.2), many other tasks call for volunteers. Be it documentation and internationalisation endeavours, user support on mailing lists and on IRC channels (see crefmailinglists and chapter 10.4.3 respectively), or simply fixing bugs and providing ideas on how to improve the distribution, a single developer can take on as many responsibilities as desired. It is important to realise that these responsibilities are by choice as nobody within the project is in a position to order a developer to do something; not even the project leader.

While the individual developer may be perceived to be at the very bottom of Debian's organisational structure, the developer majority can overrule any organisational body within Debian's structure, including the project leader, by way of general resolution. General resolutions are Debian's primary means of reaching a consensus on non-trivial decisions. A general resolution may be proposed by anyone and will be opened for vote when enough seconders back up the proposer's call. Every developer may participate in the vote, and if a majority is reached[36], the resolution takes effect immediately and cannot be overruled (except by means of another general resolution).

The Debian users

The Debian users and the community they form are unquestionably the most important entity for the project. The Social Contract defines its users as the project's top priority (see appendix E). Debian's main purpose is thus to meet the requirements of its user base. Without its users, the project would not have a purpose. Every developer is also a user, but a critical mass of "normal users" is paramount for the Debian system to stay universal and competitive (see chapter 2.2.1).

[36] Debian uses the Condorcet method to allow its developers to voice preferences rather than simple votes; see http://en.wikipedia.org/wiki/Condorcet_method

Even though normal users cannot cast votes, they can influence the direction of the project in a plethora of ways, for instance by participating in discussions on mailing lists (see chapter 10.4.1), filing bug reports (see chapter 10.6), or stepping forward in a proactive fashion to fix problems the way they would like them fixed, before someone else gets around to it.

Ultimately, every Debian user may become a Debian developer after establishing trust and demonstrating familiarity with the rules and proceedings of the projects, as well as skills pertaining to Debian's package management system and tools (see chapter 2.5.2).

SPI — Software in the Public Interest

Without being a part of the Debian project, SPI[37] is part of its organisation. SPI was founded by Bruce Perens to act as a legal entity of the Debian project. In as such, it holds all trademarks for Debian, owns all of its monetary and material assets, and represents the project in legal matters. In addition, SPI embodies an economic entity and can accept tax-deductible donations for the project, at least within the United States of America. SPI has no authority over decisions cast within the Debian project. Along the same lines, Debian claims no authority over SPI other than over the use of Debian property held and managed by the SPI.

2.4.2 Social aspects of the community

The community behind the Debian project is similar to the communities of other comparable projects. Nevertheless, the Debian community has a very strong reputation, which makes it stand out at times. Debian has the largest developer base of all free software distributions, and it constitutes more of a meta project than a simple project, as it consists of a plethora of subprojects whose only common ground in many cases is that they are part of the Debian system. While other projects of comparable size or of comparable diversity exist, Debian is unique in the combination of the two and in the way it organises the production of the Debian system. Those who support the community commonly describe it as unparalleled in terms of it dynamics and competence, and many rank the level of support available from the community as the most important factor in choosing Debian. At the same time, the Debian community is often badmouthed as arrogant and too idealistic for the real world.

It is not the purpose of this book to argue either position. The only way to decide is to get involved with the community and see for yourself. If you are new to the project, it is probably a good idea to get to know the community before it gets

[37] http://www.spi-inc.org

to know you. In the following, I put together some useful facts about the Debian community[38].

First of all, it is important to realise that the community is made up of volunteers. Developers typically work for the project because they use the Debian operating system, believe in its philosophy, and/or otherwise enjoy contributing to a cooperative effort without commercial strings attached, and without anyone ordering them around[39]. The same applies to regular users who contribute to the community. The community welcomes anyone who keeps that in mind, and abides by the usual rules of decency and etiquette expected from people sharing common grounds (even if those grounds are virtual).

Another important point to consider is that members of the community (or "users", for short) try to be helpful whenever possible. Nevertheless, nobody likes to do someone else's work. Thus, you are unlikely to find answers without doing your part first, which means using the available resources before approaching other people for help (see chapter 10). If it is clear from the details you provide that you have thought about the problem, tried to contain it, and gone to search in other places before asking for support, people will gladly offer assistance. Maybe the best way to find answers is by asking the right questions in the right way, and of the right people. Eric S. Raymond has put together a delightful piece on how to ask smart questions[40], which I suggest you read. Having read and understood the text, you need not be afraid of asking stupid questions in a Debian forum.

A nice trait of the Debian community is the equality with which everyone is treated. For new users and veterans alike, the primary focus of attention is the problem at hand[41]. This also means that the Debian community does not consist of a crowd of needy users and a bunch of gurus that answer questions. Instead, everyone is encouraged to partake and provide helpful advice, and people do. On the one hand, people like to show off their knowledge, and if it is helpful to others, all the better. On the other hand, following the community discussions and pitching in advice here and there has proven to be an excellent way to learn more about the operating system, and is advocated as such by many users[42].

[38] Please keep in mind that I am not claiming that the Debian community is unique. Much of what follows is equally applicable to other projects. I am just introducing the Debian community and laying out the facts.

[39] It is not that being told what to do is inherently bad, but it often causes quality to give way to market or time pressure, and working on products without the ability to maintain a quality level can be painful.

[40] http://www.catb.org/~esr/faqs/smart-questions.html

[41] Debian would not be Debian if long-standing users did not continuously pull each other's legs and exchange witty and sarcastic comments. However, the problem at hand maintains top priority and after bashing one another for a bit, Debian folks are usually quick to come up with helpful advice.

[42] These two patterns of behaviour are quite common in the open source world. For a complete analysis of their evolution and motivations, I refer you to Eric S. Raymond's book *The cathedral and the bazaar* (http://www.catb.org/~esr/writings/cathedral-bazaar). Another good read for those truly interested is *Understanding Open Source Software Development* by Feller and Fitzgerald (http://opensource.ucc.ie/uossd).

In the community forums (see chapter 10.4), you will find a healthy balance of technical discussion (most), humour (some), unrelated topics (few), and flamewars (rare)[43]. Overall, interactions within the community generally have a high level of productivity, and interesting discussions sometimes ensue once a solution has been found. Frequently, these discussions culminate in improvements made to the system, either on the technical or usability side, or in the documentation. The Debian policy, which will be discussed in great depth in chapter 5.7, plays an important role in the community. The policy standardises the organisation of files and the logic of administrative processes to allow the large developer collective to provide an integrated system rather than an aggregation of different packages. At the same time, the policy also makes sure that the Debian system stays the same across all installations, independent of architecture and minor version. Everyone works with the same tools on the same ground, and solutions can be found rapidly.

To sum this up, the community is made up mostly of users who are members by choice and contributors by conviction. Newcomers are often very eager to help others in an attempt to give back to the community. Others acknowledge their lack of experience with software development and contribute towards the documentation or the maintenance of web pages. The cooperative development found in and around the Debian operating system provides a prolific basis for hobbyists, enthusiasts, and professionals alike to work towards a common goal.

2.4.3 Social aspects of the group of developers

The Debian organisation is very open and flat. Apart from the officers, delegates, and a few other privileged positions, everybody in the developer team has equal rights. The prime example is the openness of the BTS and the Debian archive under equal terms to every single developer. Should one developer neglect a certain package, then another can simply take over and provide a fixed version for official inclusion in the archive without the need to acquire special privileges. When a maintainer is temporarily unavailable, other developers can provide Non-Maintainer Upload (NMU)s to be acknowledged upon return by the maintainer. Note that NMUs are constrained to single, small fixes and cannot be used to push a new upstream release, for example. In any case, when an NMU is made, the package's official maintainer is still in control, and may opt to reverse a fix. Unless such an issue causes harm to the project and thus needs to be resolved by the technical committee, a maintainer's decision is final and irrefutable.

Should a package maintainer's unavailability extend beyond an acceptable period, a developer can announce an intent to take over a package, and claim responsibility for the package if nobody objects. This developer then becomes the new maintainer and custody of the package is fully transferred. The previous maintainer has

[43] <humour>If you want flamewars, come to the developer forums, such as the **#debian** IRC channel on **irc.debian.org**! People seem to love flamewars there.</humour>

then lost privileged influence over the package's maintenance, but frequently the involved parties coordinate upon return of the absentee to keep everybody happy.

In any case, it remains the maintainer's responsibility to see to a package's well-being, and to decide on its fate. Within a community of about a thousand developers, many different interpretations of responsibility are inevitable. The core of the project is carried by a smallish number of developers that take Debian very seriously and perform their duties with the same amount of care and devotion (or even greater) as their regular jobs, or true hobbies. The majority of Debian project members treat Debian as secondary to their main life but do their best to properly attend to the responsibilities they have chosen. A number of people have taken on too many responsibilities to be able to properly address them individually, and a few seem to interpret Debian developer status merely as the right to own a Debian email address and otherwise neglect their responsibilities.

Such variety exists in every volunteer organisation of reasonable size. Debian's approach to the coordination of the developers on the path that the project has chosen is to decentralize privileges, as mentioned previously. The openness and focus of self-organisation prevents deadlocks and guards against stalls induced through negligence by single developers.

Debian – a bazaar of cathedrals

Most packages are still maintained by individuals rather than a group of people. In as such, the Debian project can be described as "a bazaar of cathedrals[44]". Although Debian's openness prevents many of the problems described by the cathedral model, not all the issues are properly addressed. When package maintenance is handled by an individual who also maintains the jurisdiction over the package, a situation not too different from the dreaded vendor lock-in surfaces. Even though it is always possible to make suggestions or provide patches via the BTS (see chapter 10.6), the package's maintenance depends on the single maintainer, which is never a good thing.

A number maintainers have taken the lead and moved the maintenance of their packages to public, collaborative platforms, such as **Alioth**[45] to reduce such dependencies. Whether developed entirely collaboratively or not, many packages list co-maintainers: developers allowed to provide new versions of a package without having file them as NMUs.

[44] I read this description on IRC, but did not note its author at the time. It references Eric S. Raymond's definitive account of the dynamics of open source projects[42].

[45] Alioth is Debian's open source development coordination forum, based on GForge: **http://alioth.debian.org**

Debian – a meritocracy

A few task domains within the Debian project, including the responsibilities of the delegates and officers, remain restricted to a small set of people. Examples are the maintenance of the Debian archive, the management of Debian user accounts, write access to the security archive, and the administration of Debian's infrastructure. Even though every developer can theoretically take part in these tasks and request the necessary privileges, in practice, such positions undergo very few fluctuations.

While the Debian project tries to prevent these jobs from being the domain of single persons by encouraging teams, the effectiveness of such teams depends on the skill levels of its members. Certain tasks, such as the administration of Debian machines, require much experience from their caretakers, and individuals lots of enthusiasm but little experience would do more harm than good.

Within the project, the occupants of positions with greater privileges are chosen on the basis of their abilities and their achievements. Therefore, the form of government that comes closest to the organisation of the Debian project is a meritocracy[46]. To be able to rise in the Debian hierarchy, an individual must have displayed competence and contributed significantly to a domain before being chosen to occupy a position within this domain.

Among the most prominent examples here is membership of the security team. As the security team's work definitely constitutes a core component of the Debian project and the stability of the Debian system, some individuals consider it prestigious to be part of the security team. In lengthy emails they explain their great ideas and elaborate their promises of how they would be an asset to the team. Even though the security team is rather understaffed, such requests are not honoured. Instead, those promoted to the security team have made valuable contributions, helping the team without actually being part of it. Such achievements then serve as the basis for the team to evaluate the individual's ability and decide on possible membership of the team.

In discussions among Debian developers, the term "cabal" may come up from time to time. While more fictitious than a real, the term cabal refers to a group of developers with elevated priviliges or senior status within the project where nobody has the membership details. The term is used mostly jokingly, but may occasionally pop up in criticisms, usually hand in hand with an expressed desire for more openness with some of the project's internal processes. It is a contagious term which is best avoided to prevent insulting people. Cabal members are often said to be unapproachable by others; in most cases this is simply a function of being approached by too many people at once, or of being overloaded with work. The best recipe to deal with alleged cabal members is to be proactive: make sure you have read

[46]The Merriam-Webster dictionary defines a meritocracy as "a system in which the talented are chosen and moved ahead on the basis of their achievement."

the available documentation and prepare concrete questions or proposals which to present to the developers.

Trust among developers

Until recently, Debian was the only operating system that was purely community-driven. While some people like to see Debian as an instance of socialism in action[47], several parts of the project and its infrastructure require legal owners, responsible persons, and proper accountability. SPI shields the project from much of the administrative burdens surrounding an institution of the size as Debian, but many aspects remain that require developers to step in to take responsibilities. Because most of the developers have never met each other, except on mailing lists or IRC[48], it seems surprising that vital parts of the project are laid in the hands of volunteers without much ado.

For instance, the responsibility of managing the Debian infrastructure, and especially the build daemons lies solely in the hands of individuals. Effectively, these individuals ensure the integrity of the Debian archive. The Debian project exists to maintain the Debian archive, and thus the project rises and falls with the propriety of its developers. Here, too, the social dynamics of open source projects, which Eric S. Raymond describes in his book, play an important role. When it comes down to it, people have little incentive to be trustworthy (in general), and there is no profit motive in contributing to Debian. However, people do work for more than immediate gain, and praise and respect amongst peers seem to be the major driving forces behind open source projects, in this case keeping the responsible developers on track.

Similarly, the Debian project does not have a structured funding infrastructure, yet a plethora of users donate money to the project. In some cases, registered organisations have volunteered to shoulder the financial administration and to accept donations for Debian for a specific part of the globe. For most countries, however, Debian does not have such dedicated legal bodies and banking fees for international money transfers are exorbitant. In these cases, developers step in to fill the gap. The sums of money which these people handle for Debian are relatively small, but there are no constraints or contracts. If a trustee decides to abandon ship and throw a party with Debian's funds, the Debian project will have little on its side to prosecute the offender[49].

[47] "Everyone bakes a cake and everyone gets a piece…" Davor Ocelić comments: "The point actually gets deeper to a technical side too, as I see it. Current computing power is too great (and develops too fast), and lifetime is too important to any of us to waste time reinventing stuff and making the same mistakes again. Writing software today only pays off with a free software license, because you are giving it a potential to last. This is simply the professionalism of the 'new age'."

[48] IRC is the Internet Relay Chat, a worldwide chat system where users can meet in pertinent channels for discussion. See chapter 10.4.3

[49] No case of such thievery has come to my attention since I joined the Debian community.

Large parts of the Debian project work on the basis of trust alone. At times, the lack of professional management has caused serious grief (not because of misappropriation, but rather accountability), but most of the time, the trust model has served Debian well. A serious offence or abuse of the rights results in the immediate and typically irrevocable expulsion from the developer team. Apparently, Debian developer status is enough of a reason to do no harm to the project. In addition, Debian developers are non-anonymous; harming the project would seriously tarnish their reputation throughout the open source community. Moreover, given the visibility of open source software development through popular search engines, it is likely that any mischief makes the rounds even beyond the community: a previous employer once confronted me with instances of good and bad conduct I had exhibited on mailing lists and expected me to justify my behaviour or commended me on my actions.

Identification

Anonymity does not exist within the Debian developer team. Identification of developers in cyberspace is handled with GNU Privacy Guard (GPG) keys. Debian developers are free to sign their Debian-related email with a strong, cryptographic signature (and users are encouraged to do the same) for important matters. Uploads to the Debian archive must be authenticated with a signature by a current developer, and signatures are required for other organisational processes, such as voting.

GPG keys are created using software such as **GnuPG**, which is freely available. It is important to realise that the identity information, such as name and email address, are provided to **GnuPG** by the user. As a result, everyone can create keys under any name. To ensure the identity of a prospective developer, it is thus required to have a key approved by an existing developer of the Debian project. This verification requires personal contact and the consideration of an official document of identification. In chapter 2.5.2 you can find more information on the process of becoming a developer.

When developers sign each other's keys, they create a relational network known as the "Web of Trust." As one of the largest groups using digital signatures consistently, Debian forms a large portion of the global Web of Trust. A complete analysis of the trust between Debian developers is available online[50].

Thus, within Debian, every developer's real-world identity is known. While the developers (and parts of the remaining user community) usually refer to each other by their nicknames (especially on IRC), the developer's full name is publicly accessible in the developer database[51]. The developer's address and contact information are not required but are generally available, albeit only to other developers for privacy

[50] http://people.debian.org/~weasel/weboftrust/index.php
[51] http://db.debian.org

reasons. Experience has shown that real names are perceived of as being considerably more trustworthy than pseudonyms[52], and it is trust upon which the entire project is built.

Social gatherings

Social gatherings among Debian developers are quite common. With the help of online resources[51] and mailing lists, users travelling to an area frequently reach out to local Debian users and set up meetings. Usually, the "excuse" is to sign GPG keys, and then to spend many hours getting to know each other. Furthermore, at any Linux-related conference, Debian folks will get together to strengthen their personal relations. Just like the Debian project, these meetings are generally open to the public, and users (as well as other people) are welcome to join.

Debian maintains a rudimentary GPG keysigning coordination page[53]. A better coordination platform, which also supports the coordination of keysigning events and expands beyond Debian's border is Biglumber[54]. The procedure of keysigning is detailed on Debian's web site[55]. For bigger events, fully-fledged protocols exist as well[56]. Debian's **signing-party** package provides **gpg-key2ps**, which conventiently converts the key information to Postscript for printing.

2.5 Helping the Debian project

Users of the Debian system often look for ways to give back to the Debian community. The Debian project is open to everyone and people willing to help will be able to do so. In many cases, it does not matter whether a contributor is a developer or not. Accounts on colaborative platforms, such as Alioth[57] or the Debian Concurrent Version System (CVS) repository[58], can be obtained without developer status. Often, the only difference between developers and non-developers is who has the final burden of making the upload, in addition to other responsibilities that take away time.

[52] A reader of the **de.newusers.questions** newsgroup once remarked that the use of real names is favourable over pseudonyms as it allows people to concentrate on the post rather than to have to get engaged in a discussion over the sense or nonsense of these names.

[53] http://nm.debian.org/gpg.php

[54] http://www.biglumber.com

[55] http://www.debian.org/events/keysigning

[56] http://www.cryptnet.net/fdp/crypto/gpg-party.html

[57] http://alioth.debian.org

[58] http://cvs.debian.org

You will find the preferred interface for tasks at the very bottom of **aptitude**'s main selection screen. Similar to handling packages, tasks may be treated as singular entities, or unfolded to reveal the packages they suggest. If the local system is to become a SQL server, you can simply navigate to the "SQL server" task in **aptitude** as shown in figure 5.10 and hit **[+]**. Subsequently, the selection can be modified. For instance, even though **libecpg4** is considered part of a typical SQL server, it may be deselected like any other package through **aptitude**'s interface. Alternatively, a user may choose to unfold a certain task and inspect the suggested set of packages. Instead of installing the task as a whole, the user may then decide to simply install only a few of the packages the task contains. You will see that tasks in **aptitude** react just like regular packages.

It is also possible to define custom tasks by dropping task description files into **/usr/share/tasksel**[30]. Documentation on how to compose tasks is available in the **README** file installed with the **tasksel** package[31].

5.6 Package management compared

It is not the intention of this book to compare. Nevertheless, as the Debian package management system seems to be misconceived too often, it is important to establish the position of **dpkg**, **APT**, & co. within the field of automatic package management. The days have passed in which Debian's package management wiped the table clean. Today, various approaches exist, each with their own special features and annoying caveats. When people tout their favourite package management system and diss on the other available solutions, they effectively admit their own ignorance of the matter. In fact, it seems as if package management systems are more a question of faith.

Package management seems to encompass three aspects: the package format specification, the package handler, and the actual package manager. Many a Debian supporter will claim that Debian excels in all three of them. While the Debian package management tools have undeniable strengths, they are not perfect. The same can be said for the package management systems of other distributions. Thus, it is time for a quick comparison (without going into too much detail).

The basis for package management is the format of the package files themselves, which provides for a lot of the functionality. Flamewars rage with DEB supporters slashing RPM fans, and *vice versa*. A common belief among Debian supporters seems to be that the DEB format is largely superior to RPM, which is simply false (and certainly one of the reasons why Debian's reputation is not always positive). In fact, the RPM format is actually more feature-rich than DEB, but the additional

[30] Additional locations may be supported in the future, see **http://bugs.debian.org/286170**.
[31] **/usr/share/doc/tasksel/README**

features are not commonly put to use[32]. Nevertheless, in terms of the capabilities actually put to use, the two formats are just too similar to compare. The same also holds true for comparisons with other major package formats, including **pkgsrc**, **ports**, and even **.ebuilds**. Each format has its own advantages and limitations, but when it comes to package management, the administrative possibilities they support are all more or less equivalent[33].

The situation with tools handling the package files is no different. **dpkg** and the **rpm** binary are package processors (as well as other managers for other formats), provide largely the same functionality (see table 5.5): installation and removal, querying a status database, and displaying information extracted from package files; they can be told to override dependencies or disregard other rules, and they can list package contents and associate installed files with the source package. In short, what can be done with one is also possible with the other. Within each of the different implementations of package management, the boundaries between the components may shift. What counts, however, is the net result and the administrative approaches the respective toolsets enable. While **dpkg** and **rpm** and their respective package formats are fundamentally different from *e.g.* a **ports**-based system, the capabilities are more or less the same.

Table 5.5: Package handling commands available by dpkg and rpm.

dpkg	rpm
dpkg --info	rpm -qpi
dpkg --contents	rpm -qpl
dpkg --install	rpm -i
dpkg --list	rpm -qa
dpkg --listfiles	rpm -ql
dpkg --search	rpm -qf
dpkg --status	rpm -qi
dpkg --remove	n/a
dpkg --purge	rpm -E
dpkg --install --force-depends	rpm -i --nodeps
dpkg --install --force-overwrite	rpm -i --replacefiles

[32] A good example of such functionality is the concept of RPM package triggers, which allow a package to register actions to be taken when another package is manipulated and thus go beyond the standard installation scripts. Another example is that RPM allows dependencies to be met by files installed on the local filesystem. While this practice is somewhat reminiscent of the dynamic library handling which gave "dependency hell" its name, it can be useful at times.

[33] A qualitative comparison is available online: **http://www.kitenet.net/~joey/pkg-comp**

The third component of a package management system is the package manager itself, which builds upon the package manager and the format specification. For a long time, APT enjoyed unrivaled precedence in this field, but the other distributions have been busy. Nowadays, tools like **up2date**, **yum**, **urpmi**, and **emerge** are hardly behind in the amount of functionality they provide (see table 5.6), and even though APT does seem to stand out in terms of maturity and robustness, it will not be long until the others are viable alternatives.

The following table attempts to list corresponding commands of the four major automatic package managers. Please note that the comparison is APT-centric, and intended to serve more as a reference than as an argument to bash the other commands, which can each do things that APT cannot. It thus primarily serves as a map to help you distinguish between the different APT commands. I purposely do not provide a map of other managers' commands to APT because APT has all the features you need for the Debian Way of package management. Concepts and approaches available with other managers but not supported by APT are unlikely to be useful on a Debian system.

Table 5.6: Package management commands of major package management systems compared.

APT	yum	up2date	urpmi
apt-cache search	yum search	http://rpmfind.net	urpmq
apt-cache show	yum info	http://rpmfind.net	urpmq -i
apt-cache showpkg	n/a	http://rpmfind.net	n/a
apt-cache depends	n/a	n/a	n/a
apt-cache rdepends	n/a	n/a	n/a
apt-get install	yum install	up2date -i	urpmi
apt-get install --download-only	yum --download-only	up2date -d	n/a[34]
apt-get remove	n/a	n/a	n/a
apt-get remove --purge	yum remove	rpm -e	urpme
apt-get update	n/a	n/a	urpmi.update -a
apt-get upgrade	yum update	n/a	n/a
apt-get dist-upgrade	yum --obsoletes update	up2date --update	urpmi --auto-select
apt-get source	n/a	up2date --src	n/a
apt-get build-dep	n/a	n/a	n/a
apt-file search	yum provides	http://rpmfind.net	urpmf

[34] urpmq --sources [...] |xargs wget

Lastly, APT is not specific to Debian. As part of Debian's commitment to the free software community, APT is publicly available and has been ported to various package formats (most notably RPM). It is already actively being used by other distributions, including Mac OS X (Fink) and Fedora.

Comparing package management systems across Linux distributions, we reach the conclusion that all major players in the field are mere mortals. But there is more to the Debian system than the aforementioned package management utilities. Those who rank APT as the true strength of the Debian operating system are wrong. The real reason is well removed from the user interfaces, deep inside the Debian system, omnipresent, but hardly noticeable.

5.7 Power from within: the Debian policy

A group of musicians does not make an orchestra. If the goal is a symphony, it does not help if each does their own thing, or small groups form to play different pieces. If the artists are willing, a little patience can lead to acceptable results, but a true symphony requires order. For an orchestra to successfully convey the energy of a musical masterpiece, it requires individual skill, a score, a conductor, and endless hours of practice.

Introducing the Debian Symphonic Orchestra

The Debian system is not unlike a symphony: the musicians are the developers who prepare numerous packages for installation on the system. If developers simply create packages to their own liking, synergy cannot emerge. Therefore, the developers have agreed on a set of rules by which to abide, just like the members of an orchestra agree on a score to follow. Within the Debian system, the role of the conductor is taken by the package management tools, which, as shown in chapter 5.3 and chapter 5.4, observe certain rules and ensure that packages harmonise. The rules as well as the tools have been around for years, and developers have had ample time to practise their use, and to correct problems.

To continue the example, the score played by the Debian developers and observed by the package management tools is the Debian policy[35]; without the policy, the Debian distribution would be Just Another Linux. But it is not. The policy is the soul of the Debian system, it is its throbbing heart, it is the reason why Debian can put the same tools to better use than others. The policy is Debian's cookbook, with years of scrutiny perfecting each single recipe.

[35] http://www.debian.org/doc/debian-policy

7.2 Security updates

When a security problem is found in a software published as part of the Debian *stable* release, the Debian security team strives to release a security update in a competetive time frame. Depending on the severity of the problem, the team might independently attempt to patch the software or search for a solution or a workaround in an effort to provide the Debian user base with an update as soon as possible. If such a solution can be found, it is made available to the general public allow others to profit from the work of the security team. In most situations, however, security fixes are found in close cooperation with the software authors and other operating system publishers.

As a security fix becomes available, the update can be quickly uploaded to the *unstable* release and subsequently enjoys privileged propagation into the *testing* archive, usually taking no longer than two days. Nevertheless, the new version will not be able to enter the *stable* release until the next official Debian version is published. To accomodate security updates, the Debian security team operates a separate APT archive for security updates to software in the *stable* archive, hosted at security.debian.org.

Backporting security fixes

It is often the case that an upstream author fixes a security problem in a new release of the software and does not bother with previous versions. After all, previous versions are mostly considered obsolete and it would be a tedious job for a developer of a programme or a library to fix every version released with respect to every bug discovered. New versions often provide new functionality or drop old features, and it is not rare for such updates to contain more bugs. Therefore, providing the new version of the software via the security archive could cause massive problems on users' machines; some users may rely on features that were dropped in the new release, others may be negatively affected by bugs introduced with new features.

Debian *stable* is stable and new versions are not allowed to enter it. Rather than solving security problems by pushing fixed packages of newer versions into the security repository, the security team analyses the solution and backports it to the version available in the *stable* archive. This results in a package providing fixed software, without adding or dropping features. To help avoid a bug fixes introducing new bugs, fixes are restricted to the bare essentials and carefully audited.

Please keep in mind that only the *stable* release receives this special treatment. Neither the *testing* nor the *unstable* archive contents are supported by the security team. If system security is one of your primary concerns, you ought to stay with Debian *stable*. Along similar lines, only the *main* archive receives full attention. While packages in *contrib* and *non-free* are not actively ignored, they are

treated with a lower priority than software in *main*, even if a bug's severity may be greater.

Special version numbers

As a consequence of Debian's security policy, some software on a Debian *stable* system may have version numbers that suggest a vulnerability in the installed software. For instance, libssl 0.9.6 is subject to a denial of service attack, a problem which was fixed in version 0.9.6l[6]. Nevertheless, the version available on *woody* these days is 0.9.6c-2.woody.7, which appears to be an older version than 0.9.6l, and thus might alert the careful administrator that the vulnerability persists. Investigating the changelog.Debian.gz file reveals that a prior release, 0.9.6c-2.woody.5, has dealt with the problem:

```
openssl (0.9.6c-2.woody.5) stable-security; urgency=high

  * Non-maintainer upload by the Security Team
  * Apply upstream patch to fix NULL pointer dereference in
    do_change_cipher_spec (CAN-2004-0079)

[...]
```

You should therefore never rely on the version numbers when assessing the vulnerability of installed software on a *stable* system. Generally, a version number suffix such as .woody.5 suggests the influence of the security team, and thus the presence of out-of-line security updates When *woody* became *stable*, openssl was at 0.9.6c-2, so the security team appended the codename and an incremental counter for each security update (see chapter 5.7.5). The changelog.Debian.gz file, which *every* package provides under /usr/share/doc/<package> will help to resolve any ambiguities.

The Debian security archive

Security updates are published in the Debian security archive, which is separate from the official Debian archive with the *stable* release among other things. Updates can be fetched and installed from this archive in one of two ways, depending on your needs.

Manual verification and installation of updates

If you have high security requirements, the best way of obtaining and installing the fixes is to download them manually from the locations specified in the DSA

[6] http://www.debian.org/security/2004/dsa-465

announcing the fixed packages. For each file, the announcement specifies the MD5 checksum and size. Prior to installation, the downloaded file must be verified against this data. The announcement itself is cryptographically signed by a member of the the Debian security team to verify the data's integrity.

For instance, DSA 588 contained the following link for the i386 architecture:

```
[...]
http://security.debian.org/pool/updates/main/g/gzip/gzip_1.3.2-3woody3_i386.deb
  Size/MD5 checksum: 62076 536b666d29bcc648a1f105b3e5ef0708
[...]
```

The procedure for verifying and installing the file is as follows:

```
~$ cd /tmp
~$ wget http://security.debian.org/[...]/gzip/gzip_1.3.2-3woody3_i386.deb
~$ wc -c gzip_1.3.2-3woody3_i386.deb
62076  gzip_1.3.2-3woody3_i386.deb
~$ md5sum gzip_1.3.2-3woody3_i386.deb
536b666d29bcc648a1f105b3e5ef0708  gzip_1.3.2-3woody3_i386.deb
```

If the size and checksum verify correctly, the package can be installed with dpkg (see chapter 5.3.2).

The manual method of installation and verification is expected to become obsolete with the introduction of APT 0.6 (see chapter 7.5.2).

Automatic security updates with APT

If your security requirements are not so strict, you might prefert the more convenient approach of having APT automatically download and install these updates. Obviously, this relies on you checking for and installing updates on a regular basis (or with automatic tools, see chapter 5.11.4).

To enable APT to install packages from this archive, the following repository must be added to /etc/apt/sources.list. Instead of the canonical name *stable*, you may want to hardcode the current release in its place, as shown in the commented line. See chapter 4 for more information on this distinction.

```
deb http://security.debian.org sarge/updates main contrib non-free
```

Contrary to some rumours, the security team treats all Debian architectures (see chapter 4.5) equally. Therefore, security updates are available for all Debian architectures at roughly the same time. The Debian build infrastructure provides special, prioritised handling of security-related package updates.

The security archive is not officially mirrored, although unofficial mirrors exist on various servers. Instead, the Debian security servers are designed to handle massive

quantities of requests[7]. A security update must be available to all users at the earliest possible moment. The Debian mirrors use a pull strategy to stay synchronised, and an update may take up to 24 hours to propagate to the primary servers, and even longer to make it to the secondary mirrors. Debian security updates are critical updates which must be made available at the earliest possible moment. Therefore, the mirror propagation times cannot be tolerated and security updates are served from a single location. Unofficial mirrors of the security archive exist nevertheless.

With the above two lines in **/etc/apt/sources.list**, keeping a Debian system secure is as easy as

```
~# apt-get update && apt-get --show-upgraded upgrade
```

The **--show-upgraded** option is not necessary but good practice. It will cause **APT** to display a list of things it plans to do, and ask the administrator whether it is okay to proceed. In fact, I suggest you make **APT** use this flag by default by adding **APT::Get::Show-Upgraded true** to **/etc/apt/apt.conf** (see chapter 5.4.2).

7.3 Security out of the box

Debian does not try to be as secure as possible (like *e.g.* the OpenBSD project). Instead, the Debian operating system constitutes a balance between ease of administration and security. Nevertheless, Debian is secure enough for most purposes, and its administrative paradigms make it even more applicable in situations where the security of a system is an important factor.

As the software distributed with the operating system comes mostly from external sources, a Debian system can only ever be as secure as the weakest software component installed. With this in mind, it is easy to appreciate the significance of a proper installation of a Debian system (see chapter 3.2). Rather than trying to install everything the user could possibly ever want, the idea behind installing a Debian system is to install a minimal system with respect to the requirements it should address. There is no reason to run a graphical environment on a web server, and a mail server does not need to provide DNS services. By keeping the number of installed packages low, administrators can effectively lessen the chance of a security problem affecting the system. As shown previously (chapter 5.7.3), the dependencies declared by Debian packages are chosen in that spirit and only pull in packages that are absolutely required.

[7] Since the arson at the University of Twente, which destroyed Debian's security server (**http://lists.debian.org/debian-devel-announce/2002/11/msg00009.html**), a more redundant setup has been put in place