Debian Developer's Reference

Release 14.18

Developer's Reference Team

2026-11-02

Inhaltsverzeichnis

1 Geltungsbereich dieses Dokuments **3**

2 Bewerbung auf eine Mitgliedschaft **5**
2.1 Erste Schritte . . . 5
2.2 Debian-Mentoren und -Sponsoren . . . 6
2.3 Registrierung als Debian Mitglied . . . 6

3 Pflichten von Debian-Entwicklern **9**
3.1 Pflichten von Paketbetreuern . . . 9
3.1.1 Auf die nächste `Stable`-Veröffentlichung hinarbeiten . . . 9
3.1.2 Pakete in `Stable` betreuen . . . 9
3.1.3 Verwalten Release-kritischer Fehler . . . 10
3.1.4 Abstimmung mit Originalautoren . . . 10
3.2 Verwaltungspflichten . . . 10
3.2.1 Verwaltung Ihrer Debian-Informationen . . . 11
3.2.2 Verwalten Ihres öffentlichen Schlüssels . . . 11
3.2.3 Abstimmungen . . . 11
3.2.4 Entscheidungsfindung . . . 11
3.2.5 Elegant Urlaub machen . . . 12
3.2.6 Sich zurückziehen . . . 12
3.2.7 Nach dem Ausscheiden zurückkehren . . . 13

4 Ressourcen für Debian Mitglieder **15**
4.1 Mailinglisten . . . 15
4.1.1 Grundregeln für die Benutzung . . . 15
4.1.2 Haupt-Entwickler-Mailinglisten . . . 15
4.1.3 Spezielle Listen . . . 16
4.1.4 Antrag auf neue entwicklungsbezogene Listen . . . 16
4.2 IRC-Kanäle . . . 16
4.3 Dokumentation . . . 17
4.4 Debian-Maschinen . . . 17
4.4.1 Der Bug-Tracking-Server . . . 17
4.4.2 Der FTP-Master-Server . . . 18
4.4.3 Der WWW-Master-Server . . . 18
4.4.4 Der people-Webserver . . . 18
4.4.5 salsa.debian.org: Git Repositorys und kollaborative Entwicklungsplattform . . . 18
4.4.6 GitHub.com: Senden von Pull-Anfragen an Upstream-Repositorys . . . 18
4.5 Die Entwicklerdatenbank . . . 19
4.6 Das Debian-Archiv . . . 19
4.6.1 Bereiche . . . 21
4.6.2 Architekturen . . . 21

4.6.3 Pakete . . . 21
4.6.4 Distributionen . . . 22
4.6.4.1 Stable, testing und unstable . . . 22
4.6.4.2 Weitere Informationen über die Testing-Distribution . . . 23
4.6.4.3 Experimental . . . 23
4.6.5 Codenamen der Veröffentlichungen . . . 23
4.7 Debian-Spiegel . . . 24
4.8 Das Incoming-System . . . 24
4.9 Paketinformationen . . . 25
4.9.1 Im Web . . . 25
4.9.2 Das Hilfswerkzeug `dak ls` . . . 25
4.10 Das Debian-Paketverfolgungssystem . . . 26
4.11 Paketübersicht des Entwicklers . . . 26
4.12 Debians FusionForge-Installation: Alioth . . . 26
4.13 Goodies für Debian Mitglieder . . . 26

5 Pakete verwalten **27**
5.1 Neue Pakete . . . 27
5.2 Änderungen im Paket aufzeichnen . . . 28
5.3 Das Paket testen . . . 28
5.4 Layout des Quellpakets . . . 29
5.5 Eine Distribution auswählen . . . 29
5.5.1 Sonderfall Uploads in die Distributionen `Stable` und `Oldstable` . . . 30
5.5.2 Sonderfall `stable-updates` Suite . . . 31
5.5.3 Sonderfall Uploads nach `testing/testing-proposed-updates` . . . 31
5.6 Ein Paket hochladen . . . 31
5.6.1 Source und Binär Uploads . . . 31
5.6.2 Hochladen nach `ftp-master` . . . 32
5.6.3 Verzögerte Uploads . . . 32
5.6.4 Sicherheits-Uploads . . . 33
5.6.5 Andere Upload-Warteschlangen . . . 33
5.6.6 Benachrichtigungen . . . 33
5.7 Angabe des Paketbereichs, des Unterbereichs und der Priorität . . . 33
5.8 Neue Upstream Versionen . . . 34
5.9 Fehlerbehandlung . . . 34
5.9.1 Fehlerüberwachung . . . 35
5.9.2 Auf Fehler antworten . . . 35
5.9.3 Verwaltung von Fehlerberichten . . . 35
5.9.4 Wann Fehler durch neue Uploads geschlossen werden . . . 37
5.9.5 Handhabung von sicherheitsrelevanten Fehlern . . . 38
5.9.5.1 Debian Security Tracker . . . 38
5.9.5.2 Vertraulichkeit . . . 38
5.9.5.3 Sicherheitsankündigungen . . . 39
5.9.5.4 Pakete vorbereiten, um Sicherheitsthemen anzugehen . . . 39
5.9.5.5 Hochladen eines korrigierten Pakets . . . 40
5.10 Paket-Updates abonnieren . . . 41
5.11 Verschieben, Entfernen, Verwaisen, Adoptieren und Wiedereinführen von Paketen . . . 41
5.11.1 Pakete verschieben . . . 41
5.11.2 Pakete entfernen . . . 41
5.11.2.1 Entfernen von Paketen aus `Incoming` . . . 42
5.11.3 Umbenennen oder Ersetzen von Paketen . . . 43
5.11.4 Verwaisen von Paketen . . . 43
5.11.5 Adoption eines Pakets . . . 43
5.11.6 Wiedereinführen vom Paketen . . . 44
5.12 Portieren und portiert werden . . . 44
5.12.1 Seien Sie freundlich zu Portierern . . . 44
5.12.2 Leitlinien für Uploads von Portierern . . . 46
5.12.2.1 Neu kompilieren oder rein binärer NMU . . . 46

5.12.2.2 Wann Sie als Portierer ein Quell-NMU durchführen sollten . . . 46
5.12.3 Portierungs-Infrastruktur und -Automatisierung . . . 47
5.12.3.1 Mailinglisten und Web-Seiten . . . 47
5.12.3.2 Werkzeuge der Portierer . . . 47
5.12.3.3 `wanna-build` . . . 47
5.12.4 Wenn Ihr Paket *nicht* portierbar ist . . . 48
5.12.5 Unfreie Pakete als automatisch erstellbar kennzeichnen . . . 48
5.13 Non-Maintainer Uploads (NMUs) . . . 48
5.13.1 Wann und wie ein NMU durchgeführt wird . . . 49
5.13.2 NMUs und `debian/changelog` . . . 50
5.13.3 Benutzung der Warteschlange `DELAYED/` . . . 50
5.13.4 NMUs aus Sicht des Paketbetreuers . . . 51
5.13.5 Quell-NMUs gegenüber rein binären NMUs (binNMUs) . . . 51
5.13.6 NMUs gegenüber QS-Uploads . . . 52
5.13.7 NMUs gegenüber Team-Uploads . . . 52
5.14 Pakete bergen . . . 52
5.14.1 Wann es berechtigt ist, ein Paket zu bergen . . . 53
5.14.2 Wie man Pakete birgt . . . 53
5.15 Gemeinschaftliche Verwaltung . . . 54
5.16 Die Distribution Testing . . . 54
5.16.1 Grundlagen . . . 54
5.16.2 Aktualisierungen von Unstable . . . 55
5.16.2.1 Veraltet . . . 55
5.16.2.2 Entfernen aus Testing . . . 56
5.16.2.3 Zirkulare Abhängigkeiten . . . 56
5.16.2.4 Beeinflussen eines Pakets in Testing . . . 56
5.16.2.5 Einzelheiten . . . 56
5.16.3 Direkte Aktualisierungen für Testing . . . 57
5.16.4 Häufig gestellte Fragen . . . 57
5.16.4.1 Was sind veröffentlichungskritische Fehler und wie werden diese gezählt? . . . 57
5.16.4.2 Wie kann das Installieren eines Pakets in `Testing` andere Pakete möglicherweise beschädigen? . . . 58
5.17 Das Archive von Stable Backports . . . 58
5.17.1 Grundlagen . . . 58
5.17.2 Ausnahmen von der erst-testing Regel . . . 58
5.17.3 Wer kann Pakete im Stable-Backports Archiv verwalten? . . . 58
5.17.4 Wann kann man damit beginnen Pakete nach stable-backports hoch zu laden? . . . 59
5.17.5 Wie lange muss ein Paket betreut werden wenn es nach stable-backports geladen worden ist? . . . 59
5.17.6 Wie oft sollte man ein Paket nach stable-backport hochladen? . . . 59
5.17.7 Wie kann man mehr über das Zurückportieren erfahren? . . . 59

6 Optimale Vorgehensweise beim Paketieren **61**
6.1 Optimale Vorgehensweisen für `debian/rules` . . . 61
6.1.1 Helferskripte . . . 61
6.1.2 Pakete mit mehreren Binärdateien . . . 62
6.2 Optimale Vorgehensweisen für `debian/control` . . . 62
6.2.1 Der Paketname . . . 62
6.2.2 Allgemeine Leitlinien für Paketbeschreibungen . . . 62
6.2.3 Die Paketübersicht oder Kurzbeschreibung . . . 63
6.2.4 Die ausführliche Beschreibung . . . 63
6.2.5 Homepage der Originalautoren . . . 64
6.2.6 Ort des Versionsverwaltungssystems . . . 64
6.2.6.1 Vcs-Browser . . . 64
6.2.6.2 Vcs-* . . . 64
6.3 Optimale Vorgehensweisen für `debian/changelog` . . . 65
6.3.1 Verfassen nützlicher Änderungsprotokolleinträge . . . 65
6.3.2 Auswahl Dringlichkeit des Uploads . . . 65

6.3.3 Häufige Missverständnisse über Änderungsprotokolleinträge . . . 66
6.3.4 Häufige Fehler in Änderungsprotokolleinträgen . . . 66
6.3.5 Änderungsprotokolle mit NEWS.Debian-Dateien ergänzen . . . 67
6.4 Optimale Vorgehensweisen rund um Sicherheit . . . 67
6.5 Optimale Vorgehensweisen für Betreuerskripte . . . 67
6.5.1 Generelle Richtlinien . . . 68
6.5.2 Erwartetes Verhalten von komplexen postinst Skripten . . . 68
6.6 Konfigurationsverwaltung mit debconf . . . 68
6.6.1 Missbrauchen Sie debconf nicht . . . 69
6.6.2 Allgemeine Empfehlungen für Autoren und Übersetzer . . . 69
6.6.2.1 Schreiben Sie korrektes Englisch . . . 69
6.6.2.2 Seien Sie freundlich zu Übersetzern . . . 69
6.6.2.3 Entfernen Sie die fuzzy-Markierungen in vollständigen Übersetzungen, wenn Sie Tipp- und Rechtschreibfehler korrigieren . . . 70
6.6.2.4 Treffen Sie keine Annahmen über Schnittstellen . . . 71
6.6.2.5 Reden Sie nicht in der ersten Person . . . 71
6.6.2.6 Formulieren Sie geschlechtsneutral . . . 71
6.6.3 Definition von Vorlagenfeldern . . . 71
6.6.3.1 Typen . . . 71
6.6.3.2 Description: Kurze und erweiterte Beschreibung . . . 72
6.6.3.3 Choices . . . 73
6.6.3.4 Default . . . 73
6.6.4 Gestaltungsrichtlinie für Vorlagenfelder . . . 73
6.6.4.1 Feld "Type" . . . 73
6.6.4.2 Feld "Description" . . . 73
6.6.4.3 Das Feld "Choices" . . . 74
6.6.4.4 Das Feld "Default" . . . 74
6.7 Internationalisierung . . . 75
6.7.1 Handhabung von Debconf-Übersetzungen . . . 75
6.7.2 Internationalisierte Dokumentation . . . 75
6.8 Optimale Vorgehensweisen für debian/patches . . . 75
6.9 Häufig vorkommende Paketierungssituationen . . . 76
6.9.1 Pakete benutzen autoconf/automake . . . 76
6.9.2 Bibliotheken . . . 76
6.9.3 Dokumentation . . . 76
6.9.4 Besondere Pakettypen . . . 77
6.9.5 Architekturunabhängige Daten . . . 77
6.9.6 Eine bestimmte Locale wird während des Paketbaus benötigt . . . 77
6.9.7 Machen Sie Übergangspakete deborphan-konform . . . 78
6.9.8 Optimale Vorgehensweisen für .orig.tar.{gz,bz2,xz}-Dateien . . . 78
6.9.8.1 Unberührter Quellcode . . . 78
6.9.8.2 Neu paketierter Originalquellcode . . . 79
6.9.8.3 Ändern binärer Dateien . . . 79
6.9.9 Optimale Vorgehensweisen für Debug-Pakete . . . 80
6.9.9.1 Automatisch erstellte Debug-Pakete . . . 80
6.9.9.2 Manuelle -dbg Pakete . . . 80
6.9.10 Optimale Vorgehensweisen für Meta-Pakete . . . 81

7 Jenseits von Paketierung **83**
7.1 Fehler berichten . . . 83
7.1.1 Viele Fehler auf einmal berichten (Masseneinreichung von Fehlern) . . . 84
7.1.1.1 Usertags . . . 84
7.2 Qualitätssicherungsbestreben . . . 84
7.2.1 Tägliche Arbeit . . . 84
7.2.2 Treffen zur gemeinschaftlichen Behebung von Fehlern (Bug-Squashing-Partys) . . . 85
7.3 Andere Paketbetreuer kontaktieren . . . 85
7.4 Sich mit inaktiven und/oder nicht erreichbaren Paketbetreuern beschäftigen . . . 85
7.5 Umgang mit wiederholt nachlässigen Debian-Entwicklern . . . 86

7.6 Zusammenwirken mit zukünftigen Debian-Entwicklern . . . 87
7.6.1 Pakete sponsern . . . 87
7.6.1.1 Ein neues Paket sponsern . . . 87
7.6.1.2 Eine Aktualisierung eines existierenden Pakets sponsern . . . 88
7.6.2 Einrichten von Uploadberechtigungen für DM . . . 89
7.6.3 Neue Entwickler befürworten . . . 89
7.6.4 Handhabung von Bewerbungen neuer Betreuer . . . 90

8 Internationalisierung und Übersetzungen **91**
8.1 Wie Übersetzungen in Debian gehandhabt werden . . . 91
8.2 I18N & L10N FAQ für Paketbetreuer . . . 92
8.2.1 Wie ein vorliegender Text übersetzt wird . . . 92
8.2.2 Wie eine vorliegende Übersetzung überprüft wird . . . 92
8.2.3 Wie eine vorliegende Übersetzung aktualisiert wird . . . 92
8.2.4 Wie Fehlerberichte gehandhabt werden, die eine Übersetzung betreffen . . . 92
8.3 I18n- & L10n-FAQ für Übersetzer . . . 93
8.3.1 Wie man bei Übersetzungsbemühungen helfen kann . . . 93
8.3.2 Wie eine Übersetzung zur Eingliederung in ein Paket bereitgestellt wird . . . 93
8.4 Beste aktuelle Vorgehensweise bezüglich L10n . . . 93

9 Überblick über die Werkzeuge der Debian-Betreuer **95**
9.1 Hauptwerkzeuge . . . 95
9.1.1 `dpkg-dev` . . . 95
9.1.2 `debconf` . . . 95
9.1.3 `fakeroot` . . . 96
9.2 Lint-Werkzeuge für Pakete . . . 96
9.2.1 `lintian` . . . 96
9.2.2 `lintian-brush` . . . 96
9.2.3 `piuparts` . . . 96
9.2.4 `debdiff` . . . 96
9.2.5 `diffoscope` . . . 97
9.2.6 `duck` . . . 97
9.2.7 `adequate` . . . 97
9.2.8 `i18nspector` . . . 97
9.2.9 `cme` . . . 98
9.2.10 `licensecheck` . . . 98
9.2.11 `blhc` . . . 98
9.3 Helferskripte für `debian/rules` . . . 98
9.3.1 `debhelper` . . . 98
9.3.2 `dh-make` . . . 98
9.3.3 `equivs` . . . 98
9.4 Werkzeuge zum Paketbau . . . 99
9.4.1 `git-buildpackage` . . . 99
9.4.2 `debootstrap` . . . 99
9.4.3 `pbuilder` . . . 99
9.4.4 `sbuild` . . . 99
9.5 Programme zum Hochladen von Paketen . . . 99
9.5.1 `dupload` . . . 99
9.5.2 `dput` . . . 100
9.5.3 `dcut` . . . 100
9.6 Automatisierung der Verwaltungsaufgaben . . . 100
9.6.1 `devscripts` . . . 100
9.6.2 `reportbug` . . . 100
9.6.3 `autotools-dev` . . . 100
9.6.4 `dpkg-repack` . . . 100
9.6.5 `alien` . . . 101
9.6.6 `elpa-dpkg-dev-el` . . . 101
9.6.7 `dpkg-depcheck` . . . 101

9.6.8 debputy . . . 101
9.7 Portierungswerkzeuge . . . 101
9.7.1 dpkg-cross . . . 101
9.8 Dokumentation und Information . . . 102
9.8.1 debian-policy . . . 102
9.8.2 doc-debian . . . 102
9.8.3 developers-reference . . . 102
9.8.4 maint-guide . . . 102
9.8.5 debmake-doc . . . 103
9.8.6 packaging-tutorial . . . 103
9.8.7 how-can-i-help . . . 103
9.8.8 docbook-xml . . . 103
9.8.9 debiandoc-sgml . . . 103
9.8.10 debian-keyring . . . 103
9.8.11 elpa-debian-el . . . 103

Debian Entwicklerreferenz Team <developers-reference@packages.debian.org>

- Copyright © 2019 - 2026 Holger Levsen
- Copyright © 2015 - 2020 Hideki Yamane
- Copyright © 2008 - 2015 Lucas Nussbaum
- Copyright © 2004 - 2007 Andreas Barth
- Copyright © 2002 - 2009 Raphaël Hertzog
- Copyright © 1998 - 2003 Adam Di Carlo
- Copyright © 1997 - 1998 Christian Schwarz

Dieses Handbuch ist freie Software, sie können es nach den Bedingungen der GNU General Public License wie von der Free Software Foundation veröffentlicht, Version 2 oder (nach ihrer Option) später verteilen oder verändern .

Dies wird in der Hoffnung verteilt, dass es nützlich sein wird, jedoch ohne jegliche Garantie; ohne auch nur die implizite Garantie der Marktgängigkeit oder Eignung für einen bestimmten Zweck. Weitere Informationen finden Sie in der GNU General Public License.

Eine Kopie der GNU General Public License ist in der Debian Distribution abgelegt unter /usr/share/common-licenses/GPL-2 oder im WWW auf der GNU Webseite. Sie erhalten diese es auch schriftlich bei der Free Software Foundation, Inc., 51 Franklin Street, 5. Stock, Boston, MA 02110-1301, USA.

Dies ist die Debian Entwicklerreferenz Version `14.18`, veröffentlicht am `2026-11-02`.

Wenn Sie diese Referenz ausdrucken möchten, sollten Sie die PDF-Version als Basis verwenden. Dieses Handbuch ist auch in einigen anderen Sprachen verfügbar.

KAPITEL 1

Geltungsbereich dieses Dokuments

Der Zweck dieses Dokuments besteht darin, Debian-Entwicklern und -Betreuern einen Überblick über die empfohlenen Verfahren und die verfügbaren Ressourcen zu geben.

Zu den darin diskutierten Verfahren gehört, wie man Mitglied wird (*Bewerbung auf eine Mitgliedschaft*), wie man neue Pakete erstellt (*Neue Pakete*) und wie man Pakete hochlädt (*Ein Paket hochladen*), wie man mit Fehlerberichten umgeht (*Fehlerbehandlung*), wie Pakete verschoben, entfernt oder verwaist werden (*Verschieben, Entfernen, Verwaisen, Adoptieren und Wiedereinführen von Paketen*), wie man Pakete portiert (*Portieren und portiert werden*) und wie und wann Zwischenversionen von Paketen anderer Betreuer zu erstellen sind (*Non-Maintainer Uploads (NMUs)*).

Die in dieser Referenz besprochenen Ressourcen umfassen die Mailinglisten (*Mailinglisten*) und Server (*Debian-Maschinen*), eine Erörterung der Struktur des Debian-Archivs (*Das Debian-Archiv*), eine Beschreibung der unterschiedlichen Server, die das Hochladen von Paketen akzeptieren (*Hochladen nach ftp-master*) und eine Erörterung von Mitteln, die Paketbetreuern dabei helfen, die Qualität ihrer Pakete zu gewährleisten (*Überblick über die Werkzeuge der Debian-Betreuer*).

Es sollte klar sein, dass diese Referenz nicht die technischen Einzelheiten von Debian-Paketen erörtert oder wie sie erstellt werden. Ebenso wenig wird diese Referenz die Standards einzeln aufführen, die Debian-Software erfüllen muss. All diese Informationen finden Sie im Debian Policy Manual.

Weiterhin ist dieses Dokument *nicht ein Ausdruck der formalen Debian-Richtlinien*. Es enthält Dokumentation für das Debian-System und allgemein vereinbarte gute fachliche Gebräuche. Daher ist es nicht das, was normalerweise als ein *normierendes* Dokument bezeichnet wird.

KAPITEL 2

Bewerbung auf eine Mitgliedschaft

2.1 Erste Schritte

Also, Sie haben all die Dokumentation gelesen, sind den Debian-Leitfaden für Neue Paketbetreuer durchgegangen, verstehen alles, was im Beispielpaket `hello` vorkommt und es geht Ihnen nun darum, Ihre Lieblings-Software zu debianisieren. Was müssen Sie denn nun im Detail tun, damit Sie Debian-Entwickler werden und Ihre Arbeit in das Debian-Projekt einfließen kann?

Falls Sie das nicht bereits getan haben, schreiben Sie sich zuerst in die Mailingliste `debian-devel@lists.debian.org` ein. Senden Sie hierzu das Wort `subscribe` im Betreff einer E-Mail an `debian-devel-REQUEST@lists.debian.org`. Falls es Probleme gibt, wenden Sie sich unter `listmaster@lists.debian.org` an den Administrator der Liste. Weitere Informationen über verfügbare Listen finden Sie unter *Mailinglisten*. `debian-devel-announce@lists.debian.org` ist eine Pflichtlektüre für jeden, der die Entwicklung von Debian verfolgen möchte.

Sie sollten sich einschreiben und ein mitlesen, ohne selbst Beiträge zu verfassen, bevor Sie irgendetwas programmieren. Teilen Sie mit, woran Sie arbeiten möchten, um doppelten Aufwand zu vermeiden.

Auch `debian-mentors@lists.debian.org` ist ein sinnvolles Abonnement. Lesen Sie *Debian-Mentoren und -Sponsoren* für Details. Der IRC-Kanal `#debian` kann ebenfalls hilfreich sein, siehe auch *IRC-Kanäle*.

Wenn Sie bereits wissen wie Sie zu Debian beitragen möchten, sollten Sie Kontakt mit existierenden Debian-Betreuern aufnehmen, die an ähnlichen Aufgaben arbeiten. Auf diesem Weg können Sie von erfahrenen Entwicklern lernen. Falls Sie zum Beispiel daran interessiert sind, existierende Software für Debian zu paketieren, sollten Sie versuchen, einen Sponsor zu finden. Ein Sponsor wird mit Ihnen zusammen an Ihrem Paket arbeiten und es in das Debian-Archiv hochladen, sobald er mit Ihrer Paketierung zufrieden ist. Sie können Sponsoren auf der Mailingliste `debian-mentors@lists.debian.org` finden, indem Sie dort Ihr Paket und sich selbst beschreiben und nach einem Sponsor fragen (lesen Sie bitte *Pakete sponsern* und https://wiki.debian.org/DebianMentorsFaq, um weitere Informationen über das Sponsern zu erhalten). Wenn Sie andererseits daran interessiert sind, Debian auf alternative Architekturen oder Kernel zu portieren, können Sie spezielle Mailinglisten zur Portierung abonnieren und dort nachfragen, wie Sie am besten einsteigen können. Und schließlich können Sie sich - falls Sie Interesse an Dokumentation oder Qualitätssicherung (QS) haben - den Betreuern anschließen, die bereits an diesen Aufgaben arbeiten, und Patches sowie Verbesserungen einreichen.

Eine Schwierigkeit könnte ein zu generischer lokaler Teil in Ihrer Mailadresse sein. Begriffe wie "mail", "admin", "root" und "master" sollten vermieden werden. Bitte lesen Sie https://www.debian.org/MailingLists/ für Details.

2.2 Debian-Mentoren und -Sponsoren

Die Mailingliste `debian-mentors@lists.debian.org` wurde für die Einsteiger unter den Betreuern eingerichtet, die sich um Hilfe beim Paketieren Ihrer ersten Pakete und anderen entwicklerbezogenen Themen bemühen. Jeder neu angehende Entwickler ist eingeladen, sich auf dieser Liste einzuschreiben (siehe *Mailinglisten* für Details).

Diejenigen die persönliche Hilfe bevorzugen (z.B. mittels privater E-Mail), sollten auch an diese Liste schreiben, und ein erfahrener Entwickler wird freiwillig helfen.

In addition, if you have some packages ready for inclusion in Debian, but are waiting for your new member application to go through, you might be able to find a sponsor to upload your package for you. Sponsors are people who are official Debian Developers, and who are willing to criticize and upload your packages for you. Please read the debian-mentors FAQ at https://wiki.debian.org/DebianMentorsFaqfirst.

Falls Sie ein Mentor und/oder Sponsor werden möchten, sind weitere Informationen in *Zusammenwirken mit zukünftigen Debian-Entwicklern* verfügbar.

2.3 Registrierung als Debian Mitglied

Before you decide to register with Debian, you will need to read all the information available at the New Members Corner. It describes in detail the preparations you have to do before you can register to become a Debian member. For example, before you apply, you have to read the Debian Social Contract. Registering as a member means that you agree with and pledge to uphold the Debian Social Contract; it is very important that members are in accord with the essential ideas behind Debian. Reading the GNU Manifesto would also be a good idea.

Bei der Registrierung als Mitglied werden Ihre Identität und Absichten geprüft sowie Ihre technischen Fähigkeiten überprüft. Da die Anzahl der an Debian arbeitenden Personen auf über 1000 angewachsen ist und unsere Systeme an mehreren sehr wichtigen Orten eingesetzt werden, muss Debian vorsichtig sein um nicht kompromittiert zu werden. Daher müssen wir neue Mitglieder entsprechend überprüfen, bevor wir ihnen Konten auf unseren Servern zuweisen und diese Pakete hochladen können.

Bevor Sie sich tatsächlich registrieren, sollten Sie zeigen, dass Sie kompetent arbeiten und gute Beiträge leisten. Sie können dies zeigen, indem Sie Patches an die Debian-Fehlerdatenbank senden und für einige Zeit ein Paket betreuen, das durch einen existierenden Debian-Entwickler gesponsert wird. Außerdem wird erwartet, dass Mitwirkende sich für das ganze Projekt interessieren und nicht nur ihre eigenen Pakete betreuen. Falls Sie anderen Betreuern helfen können, weitere Informationen zu einem Fehler oder sogar einen Patch bereitzustellen, dann tun Sie dies!

Registration requires that you are familiar with Debian's philosophy and technical documentation. Furthermore, you need an OpenPGP key which has been signed by an existing Debian maintainer. If your OpenPGP key is not signed yet, you should try to meet a Debian Developer in person to get your key signed. There's a Key Signing Coordination page which should help you find a Debian Developer close to you. (If there is no Debian Developer close to you, alternative ways to pass the ID check may be permitted as an absolute exception on a case-by-case-basis. See the identification page for more information.)

Falls Sie noch keinen OpenPGP-Schlüssel haben, generieren Sie sich einen. Jeder Entwickler benötigt einen OpenPGP-Schlüssel, um Pakete vor den Hochladen ins Archiv signieren und verifizieren zu können. Sie sollten das Handbuch der Software lesen und kennen welche Sie paketieren, es wird wichtige und meistens auch sicherheitsrelevanten Informationen enthalten. Es sind viel mehr Sicherheitslücken auf menschliche Fehler zurückzuführen, als auf Softwarefehler oder leistungsstarke Spionagetechniken. Lesen Sie *Verwalten Ihres öffentlichen Schlüssels*, um weitere Informationen über die Verwaltung Ihrer öffentlichen Schlüssel zu erhalten.

Debian benutzt `GNU Privacy Guard` (Paket `gnupg` Version 2 oder besser) als seinen Baseline Standard. Sie können auch eine andere Implementierung von OpenPGP benutzen. OpenPGP ist ein offener Standard basierend auf RFC 4880.

Die Schlüssellänge Deines Schlüssels muss mindestens 2048-Bit betragen, jedoch sollten 4096-Bit als Schlüssellänge bevorzugt werden (siehe https://keyring.debian.org/creating-key.html), es gibt keinen Grund, kürzere Schlüssel zu verwenden und dies wäre auch wesentlich unsicherer.

Falls Ihr öffentlicher Schlüssel nicht auf einem öffentlichen Server wie `subkeys.pgp.net` liegt, lesen Sie bitte NM Schritt 2: Identitätsprüfung. Dieses Dokument enthält Anweisungen, wie Sie Ihren Schlüssel auf öffentliche Schlüsselserver ablegen. Die "New Maintainer Group" wird Ihren öffentlichen Schlüssel auf den Servern ablegen, wenn er nicht bereits dort ist.

Einige Länder schränken den Gebrauch kryptografischer Software durch Ihre Bürger ein. Dies muss jedoch die Aktivitäten als Debian-Paketbetreuer nicht behindern, da es vollkommen legal sein kann, kryptografische Produkte für die Authentifizierung anstatt für Verschlüsselungszwecke zu verwenden. Wenn Sie in einem Land leben, in dem Kryptografie sogar für Authentifizierung verboten ist, so kontaktieren Sie bitte Debian, so dass besondere Vereinbarungen getroffen werden können.

Um sich als neues Mitglied zu bewerben, benötigen Sie einen bestätigten und aktiven Debian-Entwickler, der Ihre Bewerbung unterstützt (ein `Befürworter`). Nachdem Sie eine Weile zu Debian beigetragen haben und sich als registrierter Entwickler bewerben möchten, ist es notwendig das ein bestehender Entwickler, mit dem Sie in den letzten Monaten zusammengearbeitet haben, seine Überzeugung zum Ausdruck bringt, dass Sie sinnfoll zu Debian beitragen können.

When you have found an advocate, have your OpenPGP key signed and have already contributed to Debian for a while, you're ready to apply. You can simply register on our application page. After you have signed up, your advocate has to confirm your application. When your advocate has completed this step you will be assigned an Application Manager who will go with you through the necessary steps of the New Member process. You can always check your status on the applications status board.

Weitere Informationen finden Sie auf der Debian-Website New Members Corner. Stellen Sie sicher, dass Sie mit den erforderlichen Schritten des Prozesses für neue Mitglieder vertraut sind, bevor Sie sich tatsächlich bewerben. Wenn Sie gut vorbereitet sind, können Sie später viel Zeit sparen.

KAPITEL 3

Pflichten von Debian-Entwicklern

3.1 Pflichten von Paketbetreuern

Als Paketbetreuer sollten Sie qualitativ hochwertige Pakete bereitstellen, die gut in das System integriert sind und den Debian-Richtlinien entsprechen.

3.1.1 Auf die nächste `Stable`-Veröffentlichung hinarbeiten

Es ist nicht ausreichend, Pakete hoher Qualität in `Unstable` bereitzustellen; die meisten Anwender werden nur von Ihren Paketen profitieren, wenn Sie Teil der nächsten `Stable`-Veröffentlichung sind. Es wird daher von Ihnen erwartet, dass Sie mit dem Release-Team zusammenarbeiten, um sicherzustellen, dass Ihre Pakete enthalten sein werden.

Konkreter ausgedrückt: Sie sollten überwachen, ob Ihre Pakete nach `Testing` (siehe *Die Distribution Testing*) wandern. Wenn die Migration nach Ablauf der Testperiode nicht stattfindet, sollten Sie analysieren warum diese nicht erfolgt und darauf hinarbeiten, dies zu beheben. Es könnte heißen, dass Sie Ihr Paket korrigieren müssen (z.B. im Fall Release-kritischer Fehler oder wenn das Bauen auf einigen Architekturen fehlschlägt), aber es kann auch heißen, dass andere Pakete aktualisiert bzw. korrigiert (oder manchmal auch aus `Testing` entfernt) werden müssen, um bei einer Transition zu helfen, in denen Ihr Paket aufgrund von Abhängigkeiten involviert ist. Das Release-Team kann Ihnen einige Informationen zu den aktuellen Blockern einer bestimmten Transition geben, wenn Sie diese nicht identifizieren können.

3.1.2 Pakete in `Stable` betreuen

Die meiste Arbeit von Paketbetreuern geht in das Bereitstellen aktualisierter Paketversionen in `Unstable`, aber ihre Aufgabe bedingt auch, sich um Pakete in der aktuellen Veröffentlichung von `Stable` zu kümmern.

Obwohl von Änderungen in `Stable` abgeraten wird, sind diese dennoch möglich. Immer wenn ein Sicherheitsproblem gemeldet wird, sollten Sie mit dem Sicherheits-Team zusammenarbeiten, um eine korrigierte Version bereitzustellen (siehe *Handhabung von sicherheitsrelevanten Fehlern*). Wenn Fehler des Schweregrads "important" (oder höher) gemeldet werden, sollten Sie in Betracht ziehen, eine gezielte Fehlerbehebung zur Verfügung zu stellen. Sie können das `Stable`-Release-Team fragen, ob es eine solche Aktualisierung akzeptieren würde und dann einen `stable`-Upload vorbereiten (siehe *Sonderfall Uploads in die Distributionen Stable und Oldstable*).

3.1.3 Verwalten Release-kritischer Fehler

Sie sollten Fehlerberichte zu Ihren Paketen generell so erledigen, wie es in *Fehlerbehandlung* beschrieben wird. Es gibt jedoch eine spezielle Fehlerkategorie, auf die Sie besonders Acht geben sollten – sogenannte Release-kritische Fehler (release critical bugs/RC-Fehler). Alle Fehlerberichte mit Schweregrad `critical`, `grave` oder `serious` machen das Paket ungeeignet für eine Aufnahme in die nächste `Stable`-Veröffentlichung. Sie können daher die Debian-Veröffentlichung verzögern (wenn sie ein Paket in `Testing` beeinflussen) oder Migrationen nach `Testing` blockieren (wenn sie nur ein Paket in `Unstable` beeinflussen). Im schlechtesten Fall führen sie zum Entfernen des Pakets. Daher müssen diese Fehler so schnell wie möglich behoben werden.

If, for any reason, you aren't able to fix an RC bug in a package of yours within 2 weeks (for example due to time constraints, or because it's difficult to fix), you should mention it clearly in the bug report and you should tag the bug `help` to invite other volunteers to chime in. Be aware that RC bugs are frequently the targets of Non-Maintainer Uploads (see *Non-Maintainer Uploads (NMUs)*) because they can block the `testing` migration of many packages.

Mangel an Aufmerksamkeit gegenüber Release-kritischen Fehlern wird vom QA-Team oft als Zeichen interpretiert, dass der Betreuer verschwunden ist, ohne sein Paket ordentlich zu verwaisen. Das MIA-Team (Missing In Action Team) könnte außerdem eingeschaltet werden, was dazu führen könnte, dass Ihre Pakete verwaist werden (siehe *Sich mit inaktiven und/oder nicht erreichbaren Paketbetreuern beschäftigen*).

3.1.4 Abstimmung mit Originalautoren

Ein großer Teil Ihrer Arbeit als Debian-Betreuer wird darin bestehen, mit den Upstream-Entwicklern in Kontakt zu bleiben. Benutzer von Debian melden manchmal Fehler, die nicht spezifisch für die Debian Distribution sind, an unser Fehlerverfolgungssystem (oft BTS genannt was Bug Tracking System bedeutet). Diese Fehlerberichte sollten an die Upstream-Entwickler weitergeleitet werden, damit diese in einer zukünftigen Upstream-Version behoben werden können. Normalerweise ist es am besten, wenn Sie dies tun können, aber alternativ können Sie auch den Fehler-Übermittler bitten dies zu tun. Idealerweise bekommen Sie das Upstream-Projekt dazu sich per *subscribe* für Debian Paket Tracker Notifications einzuschreiben für ihre Software in Debian.

Obwohl es nicht Ihre Arbeit ist nicht Debian-spezifische Fehler zu beheben, können Sie dies dennoch tun, wenn Sie dazu in der Lage sind. Wenn Sie solche Fehler beheben, stellen Sie sicher, dass Sie ihre Korrekturen auch an die ursprünglichen Betreuer weiterleiten. Debian-Anwender und -Entwickler werden manchmal Patches schicken, um Fehler im Ursprungsprogramm zu beheben – Sie sollten diese Patches auswerten und an die ursprünglichen Autoren weiterleiten.

In Fällen, in denen ein Fehlerbericht nach Upstream weitergeleitet wird, kann es hilfreich sein, dass der BTS-Link-Dienst zur Synchronisierung von Bug-Reports zwischen dem Upstream- und dem Debian BTS-Dienst benutzt wird.

Falls Sie die ursprünglichen Quellen verändern müssen, um ein den Richtlinien entsprechendes Paket zu erstellen, dann sollten Sie freundlich eine Korrektur vorschlagen, die dort eingefügt werden kann, so dass Sie den Quellcode bei der nächsten Version der Originalautoren nicht ändern müssen. Ganz gleich, welche Änderungen Sie möchten – versuchen Sie ein Forken vom ursprünglichen Quellcode zu vermeiden.

Da die meisten Upstream-Projekte heutzutage git als Versionskontrollsystem verwenden, bietet git-buildpackage den einfachsten Weg um Patches in Debian so zu erstellen und zu verwalten, dass die Patches beim Upstream-Projekt eingereicht werden können. Für Details lesen Sie die git-buildpackage Handbuchseiten über die Nutzung von `pq` um `debian/patches` als Git-Commits zu schreiben und zu testen und mit dem Git-Remote `upstreamvcs` einfach Patches zu und von dem Upstream Git-Zweig zu cherry-picken.

Wenn Sie der Ansicht sind, dass die ursprünglichen Entwickler Debian oder der Gemeinschaft rund um freie Software ablehnend gegenüber stehen, könnten Sie es sich nochmal überlegen, ob Sie die Software in Debian einbringen möchten. Manchmal sind die gesellschaftlichen Kosten höher, als der Gewinn, den die Software mitbringt.

3.2 Verwaltungspflichten

Ein Projekt der Größe von Debian beruht auf einer Verwaltungsinfrastruktur, um über alles den Überblick zu behalten. Als Projektmitglied haben Sie einige Pflichten, um sicherzustellen, dass alles reibungslos läuft.

3.2.1 Verwaltung Ihrer Debian-Informationen

Es gibt unter https://db.debian.org/ eine LDAP-Datenbank, die Informationen über Debian-Entwickler enthält. Sie sollten Ihre Informationen dort eintragen und bei Änderungen aktualisieren. Stellen Sie insbesondere sicher, dass Ihre Weiterleitungsadresse für Ihre debian.org E-Mail Adresse immer aktuell ist. Das gilt auch für die Adresse, zu der Ihre "debian-private" E-Mails versendet werden, wenn Sie sich auch dort angemeldet haben.

Um weitere Informationen über die Datenbank zu erhalten, lesen Sie bitte *Die Entwicklerdatenbank*.

3.2.2 Verwalten Ihres öffentlichen Schlüssels

Be very careful with your private keys. Do not place them on any public servers or multiuser machines, such as the Debian servers (see *Debian-Maschinen*). Back your keys up; keep a copy offline. Read the documentation that comes with your software; read the PGP FAQ and OpenPGP Best Practices.

Sie müssen nicht nur dafür sorgen, dass Ihr Schlüssel sicher vor Diebstahl ist, sondern auch, dass er nicht verloren geht. Generieren Sie Ihr Widerrufs-Zertifikat und erstellen Sie eine Kopie davon (am besten in Papierform). Dies wird benötigt, wenn Sie Ihren Schlüssel verloren haben.

Falls Sie Ihrem öffentlichen Schlüssel Signaturen oder Benutzerkennungen hinzufügen, können Sie den Debian-Schlüsselring aktualisieren, indem Sie Ihren Schlüssel an den Schlüsselserver unter `keyring.debian.org` senden. Aktualisierungen werden mindestens einmal monatlich von den `debian-keyring`-Paketbetreuern bearbeitet.

Wenn Sie einen komplett neuen Schlüssel hinzufügen oder einen alten entfernen möchten, benötigen Sie den durch einen anderen Entwickler neu signierten Schlüssel. Falls der alte Schlüssel kompromittiert oder ungültig ist, müssen Sie außerdem das Widerrufs-Zertifikat hinzufügen. Falls es keinen vernünftigen Grund für einen neuen Schlüssel gibt, können die Betreuer des Schlüsselrings den neuen Schlüssel ablehnen. Details finden Sie unter https://keyring.debian.org/replacing_keys.html.

Es werden die gleichen Schlüssel-Extrahierungsroutinen angewandt, wie sie unter *Registrierung als Debian Mitglied* besprochen wurden.

Eine weiterreichende Erörterung der Debian-Schlüsselverwaltung können Sie in der Dokumentation des Pakets `debian-keyring` und auf https://keyring.debian.org/ finden.

3.2.3 Abstimmungen

Obwohl Debian nicht wirklich demokratisch ist, werden demokratische Prozesse benutzt, um das Führungspersonal zu wählen und generellen Entschlüssen zuzustimmen. Diese Prozeduren sind durch die Debian-Verfassung definiert.

Im Gegensatz zur jährlichen Wahl des Projektleiters werden keine regelmäßigen Abstimmungen abgehalten und wenn doch, dann nicht einfach so. Jeder Vorschlag wird zuerst auf der Mailingliste `debian-vote@lists.debian.org` besprochen und benötigt mehrere Befürworter, bevor der Projektsekretär die Abstimmungsprozedur in Gang setzt.

Sie müssen vor der Abstimmung nicht alle Diskussionen verfolgen, da der Sekretär mehrere Aufrufe zur Abstimmung auf `debian-devel-announce@lists.debian.org` heraus gibt (und von allen Entwicklern wird erwartet, dass sie diese Liste abonniert haben). Demokratie funktioniert nicht richtig, wenn die Leute nicht an Abstimmungen teilnehmen, daher wird allen Entwicklern empfohlen abzustimmen. Die Abstimmung wird mittels OpenPGP-signierter/verschlüsselter E-Mails durchgeführt.

Die Liste aller (früheren und aktuellen) Vorschläge ist auf der Seite Debian-Abstimmungs-Informationen verfügbar, ebenso die Informationen, wie Vorschläge gemacht und unterstützt werden und darüber abgestimmt wird.

3.2.4 Entscheidungsfindung

Neben den Aspekten unterhalb von *Abstimmungen* beschrieben ist Debian dichter an einer Do-ocracy, als an einer Demokratie. Allgemein betrachtet bedeutet dies, dass Entscheidungen von den Leuten getroffen werden, die die Arbeit machen. In den meisten Fällen sind das Paketbetreuer und DPL delegates. Öffentliche Diskussion ist:

- während ermutigt und ggf. als Beitrag zur Entscheidungsfindung geschätzt,
- nicht notwendig für eine Entscheidung,

- und kann keine Entscheidung überstimmen (dies ist nur durch eine Generalresolution möglich oder durch die Eskalation an das Debian Technical Committee).

Im Sinne einer Do-ocracy wird von Mitgliedern der Debian Gemeinschaft erwartet, dass sie in konkreter und konstruktiver Weise beitragen, anstatt von anderen zu erwarten irgendeine Art von Arbeit für sie zu leisten.

For more on the social aspects of contributing to Debian, you are strongly encouraged to read the Debian Community Guidelines at https://people.debian.org/~enrico/dcg/

3.2.5 Elegant Urlaub machen

Es ist bei Entwicklern üblich, dass es Zeiten gibt, in denen sie abwesend sind, entweder wegen geplanter Ferien oder einfach, weil sie unter anderer Arbeit begraben sind. Am wichtigsten ist, dass Sie daran denken, andere Entwickler über Ihren Urlaub zu informieren, damit diese bei Problemen mit Ihren Paketen oder anderweitigen Pflichten im Projekt die erforderlichen Maßnahmen ergreifen können.

Üblicherweise bedeutet dies, dass andere Entwickler ein NMU (siehe *Non-Maintainer Uploads (NMUs)*) Ihres Pakets durchführen dürfen, weil ein großes Problem (release-kritischer Fehler, Sicherheitsaktualisierung etc.) auftritt, während Sie in Ferien sind. Manchmal sind es auch weniger kritische Dinge, wegen derer Sie verhindert sein können, aber es ist trotzdem angemessen, andere darüber zu informieren.

Um andere Entwickler zu informieren, gibt es zwei Dinge die Sie tun sollten. Zuerst senden Sie eine E-Mail an `debian-private@lists.debian.org`, in der Sie [VAC] vor den Betreff Ihrer Nachricht schreiben[1] und die Dauer angeben, wie lange Sie Urlaub machen. Sie können außerdem einige besondere Anweisungen geben, was beim Auftreten von Fehlern zu tun ist.

Das andere was Sie tun sollten, ist sich selbst in der *Die Entwicklerdatenbank* als abwesend zu kennzeichnen (auf diese Information können nur Debian-Entwickler zugreifen). Vergessen Sie nicht, die Urlaubskennzeichnung zu entfernen, wenn Sie wieder zurück sind!

Idealerweise sollten Sie sich auf den OpenPGP-Koordinierungsseiten anmelden, wenn Sie Urlaub buchen und prüfen, ob es dort jemanden gibt, der eine Signatur benötigt. Dies ist besonders dann wichtig, wenn Leute an exotische Orte fahren, an denen es noch keine Entwickler gibt, aber Leute, die an einer Bewerbung interessiert sind.

3.2.6 Sich zurückziehen

Falls Sie das Debian-Projekt verlassen wollen, sollten Sie die folgenden Schritte einhalten:

- Verwaisen Sie all Ihre Pakete wie in *Verwaisen von Paketen* beschrieben.
- Entfernen Sie sich aus Uploader-Feldern für Co- oder Team-betreute Pakete.
- Falls Sie E-Mails über einen "@debian.org" E-Mail Alias (z.B. press@debian.org) empfangen haben und möchten aus diesem entfernt werden, öffnen Sie ein RT-Ticket für die Debian-Systemadministration. Senden Sie dazu eine E-Mail an `admin@rt.debian.org`, die irgendwo im Betreff "Debian RT" enthält und angibt, von welchen Alias Sie entfernt werden möchten.
- Bitte denken Sie daran, sich auch aus Teams zurückzuziehen, z.B. entfernen Sie sich von Team-Wiki-Seiten oder Salsa-Gruppen.
- Login into https://nm.debian.org/ and then open https://nm.debian.org/process/emeritus to request emeritus status and write a goodbye message that will be automatically posted on debian-private.

  Salsa is used as authentication provider to the NM site, in case you run into problems opening the retirement process yourself, contact NM front desk using `nm@debian.org`

Es ist wichtig obigem Prozess zu folgen, da die Suche nach inaktiven Entwicklern und das Verwaisen derer Pakete erhebliche Zeit kostet und Mühe bereitet.

[1] Der Grund hierfür ist, dass die Nachricht einfach von Leuten ausgefiltert werden kann, die keine Urlaubsbenachrichtigungen lesen möchten.

3.2.7 Nach dem Ausscheiden zurückkehren

Das Konto eines zurückgetretenen Entwicklers ist als "Emeritus" markiert, wenn dem Prozess in *Sich zurückziehen* gefolgt wurde und ansonsten als "Removed". Zurückgetretene Entwickler mit einem "Emeritus"-Konto können Ihr Konto wie folgt reaktivieren:

- Melden Sie sich unter nm.debian.org mit Ihren Salsa-Anmeldedaten an.
- Folgen Sie dem Nach dem Ausscheiden zurückkehren Wizard.
- Durchlaufen Sie den verkürzten NM-Prozess (um sicherzustellen, dass der zurückgekehrte Entwickler noch immer die wichtigen Teile von P&P – "Philosophie und Prozeduren" und T&S – "Aufgaben und Fertigkeiten" kennt).

In case you run into problems contact `nm@debian.org` for further instructions.

Zurückgetretene Entwickler mit einem "Removed" Konto müssen den NM-Prozess erneut durchlaufen.

KAPITEL 4

Ressourcen für Debian Mitglieder

In diesem Kapitel finden Sie eine sehr kurze Roadmap der Debian-Mailinglisten, der Debian-Maschinen, die Ihnen als Mitglied zur Verfügung stehen, und aller anderen Ressourcen, die Ihnen bei Ihrer Arbeit zur Verfügung stehen.

4.1 Mailinglisten

Viele Unterhaltungen zwischen Debian-Entwicklern (und Anwendern) laufen über ein weites Feld von Mailinglisten, die auf `lists.debian.org` untergebracht sind. Mehr darüber, wie Sie diese Listen abonnieren oder abbestellen können, wie Sie Nachrichten abschicken, wo Sie alte Nachrichten finden und suchen können, wie Sie Listenbetreuer kontaktieren können sowie verschiedene sonstige Informationen über Mailinglisten erfahren finden Sie auf https://www.debian.org/MailingLists/. Dieser Abschnitt wird nur die Gesichtspunkte der Mailinglisten aufzeigen, die von besonderem Interesse für Entwickler sind.

4.1.1 Grundregeln für die Benutzung

Wenn Sie auf Nachrichten auf der Mailingliste antworten, senden Sie bitte keine Kopie (CC) an den ursprünglichen Verfasser, außer, wenn dieser explizit darum bittet. Jeder, der an eine Mailingliste schreibt, sollte sie lesen, um die Antworten zu sehen.

Kreuzversand (die gleiche Nachricht an mehrere Listen senden) ist unerwünscht. Kürzen Sie, wie immer im Netz, die Zitate von Artikeln, auf die Sie antworten. Halten Sie sich bitte an die allgemeinen Gepflogenheiten beim Versand von Nachrichten.

Bitte lesen Sie den Leitfaden, um weitere Informationen zu erhalten. Die Debian Community Guidelines sind es ebenfalls wert, gelesen zu werden.

4.1.2 Haupt-Entwickler-Mailinglisten

Die Haupt-Debian-Mailinglisten, die Entwickler nutzen sollten, sind:

- `debian-devel-announce@lists.debian.org`, wird benutzt, um Entwicklern wichtige Dinge anzukündigen. Es wird von allen Entwicklern erwartet, dass sie diese Liste abonnieren.
- `debian-devel@lists.debian.org`, wird benutzt, um über verschiedene entwicklungsbezogene technische Themen zu reden.
- `debian-policy@lists.debian.org`, wird benutzt, um über die Debian-Richtlinien zu diskutieren und darüber abzustimmen.

- debian-project@lists.debian.org, wird benutzt, um über verschiedene entwicklungsbezogene, nicht technische Themen zu reden.

Es sind weitere Mailinglisten für unterschiedliche spezielle Themen verfügbar. Eine Liste finden Sie unter https://lists.debian.org/.

4.1.3 Spezielle Listen

debian-private@lists.debian.org ist eine besondere Mailingliste für private Unterhaltungen zwischen Debian-Entwicklern. Das heißt, sie sollte benutzt werden, um über Dinge zu reden, die aus irgend einem Grund nicht veröffentlicht werden sollen. Eigentlich ist es eine Liste mit geringem Umfang und Benutzer werden angehalten, debian-private@lists.debian.org nicht zu benutzen, so lange es nicht wirklich nötig ist. Leiten Sie außerdem *keine* E-Mail von dieser Liste an jemanden weiter. Es sind aus naheliegenden Gründen keine Archive dieser Liste im Web verfügbar, aber Sie können sie sehen, indem Sie Ihr Shell-Konto auf master.debian.org benutzen und in das Verzeichnis ~debian/archive/debian-private/ schauen.

debian-email@lists.debian.org ist eine besondere Mailingliste, die als Wundertüte für Debian-bezogene Korrespondenz, wie den Kontakt zu ursprünglichen Autoren über Lizenzen, Fehler, etc. oder Diskussionen über das Projekt mit anderen benutzt wird, wobei es nützlich sein könnte, dass die Diskussion irgendwo archiviert wird.

4.1.4 Antrag auf neue entwicklungsbezogene Listen

Bevor Sie eine Mailingliste anfordern, die sich auf die Entwicklung eines Pakets (oder einer kleinen Gruppe verwandter Pakete) bezieht, sollten Sie überlegen, ob die Verwendung eines Alias eventuell besser geeignet wäre. So ein Alias wird durch einen Eintrag in eine .forward-aliasname-Datei auf master.debian.org angelegt, der dann eine *Ihr-Aliasname@debian.org* E-Mail Adresse gewissermaßen übersetzt.

Falls Sie entscheiden, dass eine reguläre Mailingliste auf lists.debian.org wirklich das ist was Sie wollen – nur zu. Füllen Sie eine Anfrage aus und folgen Sie diesem HOWTO.

4.2 IRC-Kanäle

Mehrere IRC-Kanäle sind für die Entwicklung von Debian bestimmt. Sie werden hauptsächlich auf dem Open and free technology community (OFTC)-Netzwerk gehostet. Der DNS-Eintrag irc.debian.org ist ein Alias für irc.oftc.net.

Der Hauptkanal für Debian ist im Allgemeinen #debian. Dies ist ein großer Kanal für allgemeine Zwecke, auf dem Benutzer aktuelle Neuigkeiten im Inhalt finden, der von Robotern bereitgestellt wird. #debian ist für englischsprachige Nutzer. Es gibt auch #debian.de, #debian-fr, #debian-br und andere Kanäle mit ähnlichen Namen für anderssprachige Nutzer.

Der Hauptkanal für die Debian-Entwicklung ist #debian-devel. Es ist ein sehr aktiver Kanal; es werden normalerweise mindestens 150 Leute zu jeder Tageszeit dort sein. Es ist ein Kanal für Leute, die an Debian arbeiten, es ist kein Support-Kanal (dafür gibt es #debian). Das Themengebiet enthält normalerweise interessante Informationen für Entwickler.

Da #debian-devel ein offener Kanal ist, sollten Sie dort nicht über Probleme sprechen, die in debian-private@lists.debian.org diskutiert werden. Es gibt einen anderen Kanal für diesen Zweck. Er heißt #debian-private und ist durch einen Schlüssel geschützt. Dieser Schlüssel ist unter master.debian.org:~debian/misc/irc-password verfügbar.

Es gibt andere zusätzliche Kanäle, die besonderen Themen gewidmet sind. #debian-bugs wird für die Koordination von Bug-Squashing-Parties benutzt. #debian-boot wird benutzt, um die Arbeit am Debian-Installationsprogramm zu koordinieren. #debian-doc wird gelegentlich benutzt, um über die Dokumentation zu reden, wie beispielsweise über das Dokument was Sie aktuell gerade lesen. Andere Kanäle beschäftigen sich mit einer Architektur oder einer Zusammenstellung von Paketen: #debian-kde, #debian-dpkg, #debian-perl, #debian-python...

Es existieren außerdem einige nicht englische Entwicklerkanäle, zum Beispiel #debian-devel-fr für französischsprachige Leute, die an der Entwicklung von Debian interessiert sind.

Speziell Debian gewidmete Kanäle gibt es auch in anderen IRC-Netzwerken.

4.3 Dokumentation

Dieses Dokument enthält viele Informationen, die für Debian-Entwickler nützlich sind, es kann aber auch nicht alles enthalten. Die meisten anderen interessanten Dokumente sind in der Entwickler-Ecke verlinkt. Nehmen Sie sich die Zeit alle diese Verweise zu durchstöbern – Sie werden viele weitere Dinge lernen.

4.4 Debian-Maschinen

Debian hat mehrere Computer, die als Server fungieren, die meisten davon, um kritische Funktionen für das Debian-Projekt bereitzustellen. Die meisten dieser Maschinen werden für Portierungzwecke benutzt und alle haben eine permanente Verbindung ins Internet.

Einige der Maschinen stehen einzelnen Entwicklern so lange zur Verfügung, wie die Entwickler die Regeln befolgen, die in den Debian-Rechner Benutzungsrichtlinien festgelegt wurden.

Allgemein gesprochen können Sie diese Maschinen nach Belieben für Debian-bezogene Zwecke nutzen. Bitte seien Sie freundlich zu den Systemadministratoren und verbrauchen Sie nicht massenhaft Plattenplatz, Netzwerk-Bandbreite oder CPU, ohne zuerst die Zustimmung der Systemadministratoren eingeholt zu haben. Üblicherweise werden diese Maschinen von Freiwilligen betrieben.

Bitte achten Sie darauf, Ihre Debian-Passwörter und SSH-Schlüssel, die auf Debian-Maschinen installiert sind, zu schützen. Vermeiden Sie Methoden zum Anmelden oder Hochladen, die Passwörter unverschlüsselt über das Internet übertragen, wie Telnet, FTP, POP, etc.

Bitte legen Sie kein Material auf Debian-Servern ab, das keinen Bezug zu Debian hat, nicht einmal, wenn Sie eine vorrangige Berechtigung haben.

Die aktuelle Liste von Debian-Maschinen ist unter https://db.debian.org/machines.cgi verfügbar. Diese Web-Seite enthält Maschinennamen, Kontaktinformationen darüber, wer sich anmelden kann, SSH-Schlüssel etc.

Falls Sie ein Problem mit einer Transaktion auf einem Debian-Server haben und der Ansicht sind, dass die Systemverwalter über dieses Problem informiert werden sollten, können Sie die Liste offener Probleme in der Warteschlange des DSA-Teams (Debian System Administration) vom Request Tracker unter https://rt.debian.org/ prüfen (Sie können sich als Benutzer "debian" anmelden, dessen Passwort ist unter `master.debian.org:~debian/misc/rt-password` verfügbar). Um ein neues Problem zu melden, senden Sie einfach eine E-Mail an `admin@rt.debian.org` und stellen Sie sicher, dass der Betreff die Zeichenkette "Debian RT" enthält. Um das DSA-Team per E-Mail zu kontaktieren, verwenden Sie für alles, was private oder vertrauliche Informationen enthält und die nicht veröffentlicht werden soll die Adresse `dsa@debian.org`, für alles andere benutzen Sie bitte `debian-admin@lists.debian.org`. Das DSA-Team ist auch auf dem IRC-Kanal `#debian-admin` im OFTC erreichbar.

Falls Sie ein Problem mit einem bestimmten Dienst haben, der sich nicht auf die Systemadministration bezieht (wie beispielsweise Pakete, die aus dem Archiv entfernt werden sollen, Vorschläge für die Website, etc.) schreiben Sie einen Fehlerbericht zu einem "Pseudo-Paket". Informationen darüber, wie Sie Fehlerberichte versenden, finden Sie unter *Fehler berichten*.

Einige der Hauptserver werden eingeschränkt, aber die Informationen von dort werden auf einen anderen Server gespiegelt.

4.4.1 Der Bug-Tracking-Server

`bugs.debian.org` ist der anerkannte Ort für die Fehlerdatenbank, das BTS (Bug Tracking System).

Falls Sie statistische Auswertungen oder Verarbeitungen von Debian-Fehlern planen, wäre dies der richtige Ort dafür. Bitte schildern Sie indes Ihre Pläne auf `debian-devel@lists.debian.org` bevor Sie etwas implementieren, um unnötige Doppelarbeit oder vergeudete Ausführungszeit zu vermeiden.

4.4.2 Der FTP-Master-Server

Auf dem Server `ftp-master.debian.org` liegt die autorisierte Kopie des Debian-Archivs. Generell landen nach ftp.upload.debian.org hochgeladene Pakete auf diesem Server, siehe *Ein Paket hochladen*.

Dieser Server ist einer wie oben erwähnt einer der eingeschränkten Server; ein Spiegel ist auf `mirror.ftp-master.debian.org` verfügbar.

Probleme mit dem Debian-Archiv müssen generell als Fehler des Pseudo-Pakets `ftp.debian.org` oder per E-Mail an `ftpmaster@debian.org` gemeldet werden, aber sehen Sie sich auch die Vorgehensweisen in *Verschieben, Entfernen, Verwaisen, Adoptieren und Wiedereinführen von Paketen* an.

4.4.3 Der WWW-Master-Server

Der Haupt-Webserver ist `www-master.debian.org`. Auf ihm liegen die offiziellen Web-Seiten, für die meisten Neulinge "das Gesicht von Debian".

Falls Sie ein Problem mit dem Debian-Webserver finden, sollten Sie generell einen Fehlerbericht an das Pseudo-Paket `www.debian.org` senden. Denken Sie daran zu prüfen, ob bereits sonst jemand dieses Problem an die Fehlerdatenbank gemeldet hat.

4.4.4 Der people-Webserver

`people.debian.org` ist der Server, der für die eigenen Web-Seiten der Entwickler über alles Mögliche mit Bezug zu Debian benutzt wird.

Falls Sie Debian spezifische Informationen haben, die Sie im Web bereitstellen möchten, können Sie dies realisieren, indem Sie das Material im Verzeichnis `public_html` in Ihrem Home-Verzeichnis auf `people.debian.org` ablegen. Es kann über die URL `https://people.debian.org/~`*Ihre-Benutzer-ID*`/` darauf zugegriffen werden.

Sie sollten nur diesen besonderen Ort benutzen, da Datensicherungen davon erstellt werden, was auf anderen Rechnern nicht der Fall wäre.

Der einzige Grund, andere Rechner zu benutzen, ist üblicherweise, wenn Sie Materialien veröffentlichen möchten, die den U.S.-Exportbeschränkungen unterliegen. In diesem Fall können Sie Server benutzen, die sich außerhalb der Vereinigten Staaten befinden.

Senden Sie eine E-Mail an `debian-devel@lists.debian.org`, falls Sie Fragen haben.

4.4.5 salsa.debian.org: Git Repositorys und kollaborative Entwicklungsplattform

Wenn Sie ein Git-Repository für eine Ihrer Debian-Arbeiten verwenden möchten, können Sie zu diesem Zweck die GitLab-Instanz von Debian mit dem Namen Salsa verwenden. GitLab bietet auch die Möglichkeit, Merge-Requests, Wiki-Seiten, Bug-Tracker und viele andere Dienste sowie eine detaillierte Optimierung der Zugriffsberechtigungen bereitzustellen, um die Zusammenarbeit an Projekten zu unterstützen.

Für mehr Informationen besuchen Sie bitte https://wiki.debian.org/Salsa/Doc.

Jedes Debian-Paket, das auf Salsa gehostet wird, hat ebenfalls Zugang zu der Salsa-CI . Die Salsa-CI-Pipeline imitiert die Tests, die nach jedem Upload zu Debian ausgeführt werden. Aber anstatt auf die Ergebnisse warten zu müssen oder die Qualität der Debian Repositorys zu riskieren, gibt die Salsa CI direkte Rückmeldung zu allen Problemen, die Ihre Änderungen verursacht oder behoben haben.

4.4.6 GitHub.com: Senden von Pull-Anfragen an Upstream-Repositorys

Sollte ein Upstream-Repository auf GitHub.com beheimatet sein dann kann die Debian Organisation benutzt werden um einen Fork des Repository für Pull-Requests an das Upstream-Projekt erstellen zu können.

Die Organisation ist offen für alle Debian Mitglieder. Um eine Mitgliedschaft zu beantragen muss ein Issue im Debian/.github Meta Repository erstellt werden.

4.5 Die Entwicklerdatenbank

Die Entwicklerdatenbank unter https://db.debian.org/ ist ein LDAP-Verzeichnis zur Verwaltung von Debian-Entwicklereigenschaften. Sie können diese Ressource benutzen, um eine Liste der Debian-Entwickler zu durchsuchen. Ein Teil dieser Informationen ist auch durch den Dienst "finger" auf Debian-Servern verfügbar. Versuchen Sie `finger ihr-benutzername@db.debian.org`, um zu sehen, was er berichtet.

Entwickler können sich in der Datenbank anmelden, um verschiedene Informationen über sich selbst zu ändern, wie beispielsweise:

- Die Einstellungen der Weiterleitung ihrer debian.org E-Mail-Adresse als auch das Behandeln von Spam. Siehe https://db.debian.org/forward.html für eine Beschreibung aller Optionen.
- Die Anmeldung zu debian-private
- Ob Sie in Urlaub sind
- Persönliche Informationen, wie Ihre Adresse, das Land, Längen- und Breitengrad Ihres Wohnortes für die Weltkarte der Entwickler, Telefon- und Faxnummern, IRC-Nickname und Homepage
- Passwort und bevorzugte Shell auf Maschinen des Debian-Projekts

Natürlich kann nicht auf die meisten der Informationen durch die Öffentlichkeit zugegriffen werden. Lesen Sie für weitere Informationen die Dokumentation unter https://db.debian.org/doc-general.html.

Entwickler können auch ihre SSH-Schlüssel senden, die für die Authentifizierung auf den offiziellen Debian-Maschinen benutzt werden und sogar neue *.debian.net-DNS-Einträge hinzufügen. Diese Funktionen sind unter https://db.debian.org/doc-mail.html dokumentiert.

4.6 Das Debian-Archiv

Die Distribution Debian besteht aus vielen Paketen (aktuell rund `40000` Quellpakete) und ein paar zusätzlichen Dateien (wie beispielsweise Dokumentation und Images von Installationsmedien).

Hier ist ein Beispielverzeichnisbaum eines kompletten Debian-Archivs:

```
dists/stable/main/
dists/stable/main/binary-amd64/
dists/stable/main/binary-armel/
dists/stable/main/binary-i386/
     ...
dists/stable/main/source/
     ...
dists/stable/main/disks-amd64/
dists/stable/main/disks-armel/
dists/stable/main/disks-i386/
     ...

dists/stable/contrib/
dists/stable/contrib/binary-amd64/
dists/stable/contrib/binary-armel/
dists/stable/contrib/binary-i386/
     ...
dists/stable/contrib/source/

dists/stable/non-free/
dists/stable/non-free/binary-amd64/
dists/stable/non-free/binary-armel/
dists/stable/non-free/binary-i386/
     ...
dists/stable/non-free/source/
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
dists/stable/non-free-firmware/
dists/stable/non-free-firmware/binary-amd64/
dists/stable/non-free-firmware/binary-armel/
dists/stable/non-free-firmware/binary-i386/
     ...
dists/stable/non-free-firmware/source/

dists/testing/
dists/testing/main/
     ...
dists/testing/contrib/
     ...
dists/testing/non-free/
     ...
dists/testing/non-free-firmware/
     ...

dists/unstable
dists/unstable/main/
     ...
dists/unstable/contrib/
     ...
dists/unstable/non-free/
     ...
dists/unstable/non-free-firmware/
     ...

pool/
pool/main/a/
pool/main/a/apt/
     ...
pool/main/b/
pool/main/b/bash/
     ...
pool/main/liba/
pool/main/liba/libalias-perl/
     ...
pool/main/m/
pool/main/m/mailx/
     ...
pool/non-free/d/
pool/non-free/d/doc-rfc/
     ...
pool/non-free-firmware/f/
pool/non-free-firmware/f/firmware-nonfree/
     ...
```

Wie Sie sehen können, enthält das Verzeichnis auf der obersten Ebene die beiden Verzeichnisse `dists/` und `pool/`. Letzteres ist ein "Pool", in dem sich die Pakete derzeit befinden. Er wird von der Archiv-Verwaltungsdatenbank und den beiliegenden Programmen gehandhabt. Ersteres enthält die Distributionen `stable`, `testing` und `unstable`. Die Dateien `Packages` und `Sources` in den Distributions-Unterverzeichnissen können auf Dateien im Verzeichnis `pool/` verweisen. Der Verzeichnisbaum unterhalb jeder Distribution ist auf die gleiche Art angeordnet. Was im Folgenden für `stable` beschrieben wird, ist gleichermaßen auf die Distributionen `unstable` und `testing` anwendbar.

`dists/stable` enthält drei Verzeichnisse, nämlich `main`, `contrib` und `non-free` und `non-free-firmware`.

In jedem Bereich gibt es ein Verzeichnis für Quellpakete (`source`) und ein Verzeichnis für jede unterstützte Architektur (`binary-i386`, `binary-amd64`, etc.).

Der Bereich `main` enthält zusätzliche Verzeichnisse, die die Medien-Images und einige notwendige Teile der Dokumentation, die zum Installieren der Debian-Distribution auf einer speziellen Architektur (`disks-i386`, `disks-amd64`, etc.) benötigt werden.

4.6.1 Bereiche

Der Bereich `main` des Debian-Archivs ist das, was die **offizielle Debian-Distribution** ausmacht. Der Bereich `main` ist offiziell, da er alle Richtlinien vollständig erfüllt. Bei den beiden anderen Abschnitten ist dies zu einem unterschiedlichen Grad nicht der Fall. Von daher sind sie **nicht** offizieller Teil von Debian.

Jedes Paket im Bereich "main" muss vollständig die Debian-Richtlinien für Freie Software (DFSG) erfüllen sowie alle anderen Anforderungen des Regelwerks, das im Debian Policy Manual beschrieben wurde. Die DFSG sind Debians Definition von "freier Software". Prüfen Sie das Debian Policy Manual für Details.

Pakete im Bereich `contrib` müssen den DFSG entsprechen, könnten aber an anderen Anforderungen scheitern. Zum Beispiel könnten sie von unfreien Paketen abhängen.

Pakete die nicht die DFSG erfüllen, werden in den Bereich `non-free` oder `non-free-firmware` platziert. Diese Pakete werden nicht als Teil der Debian-Distribution betrachtet, obwohl ihre Benutzung ermöglicht und die Infrastruktur (wie die Fehlerdatenbank und die Mailinglisten) für unfreie Pakete bereitgestellt wird.

Das Debian Policy Manual enthält eine genauere Definition dieser vier Bereiche. Die vorhergehende Erläuterung ist nur eine Einführung.

Die Unterteilung in vier Bereiche auf der obersten Ebene des Archivs ist für all die Leute wichtig, die Debian weitergeben möchten, entweder über FTP-Server im Internet oder auf CD-ROMs: Rechtliche Risiken können vermieden werden, wenn nur die Abschnitte `main` und `contrib` weitergegeben werden. Einige Pakete im Bereich `non-free` erlauben zum Beispiel keine kommerzielle Weitergabe.

Andererseits könnte ein CD-ROM-Verkäufer die einzelnen Paketlizenzen der Pakete in `non-free` leicht prüfen und seinen CD-ROMs so viele wie erlaubt hinzufügen. (Da dies von Verkäufer zu Verkäufer stark variiert, können Debian-Entwickler diese Arbeit nicht erledigen.)

Beachten Sie, dass der Begriff Bereich auch im Bezug auf Kategorien benutzt wird, um die Organisation und das Durchstöbern verfügbarer Pakete zu vereinfachen, z.B. `admin`, `net`, `utils` etc. Diese Bereiche (eher Unterbereiche) existierten irgendwann einmal in der Form von Unterverzeichnissen innerhalb des Debian-Archivs. Heutzutage existieren sie nur noch in den "Section"-Kopfzeilenfeldern der Pakete.

4.6.2 Architekturen

In den Anfangstagen war der Linux-Kernel und somit Debian nur für die Intel-i386-Plattformen (oder höher) verfügbar. Aber als Linux zunehmend populärer wurde, wurde der Kernel auf andere Architekturen portiert und Debian begann, diese zu unterstützen. Und als ob die Unterstüzung so viel verschiedener Hardware noch nicht genug wäre, entschied Debian, einige Portierungen zu erstellen, die auf anderen Unix-Kerneln beruhten, wie `hurd` und `kfreebsd`.

Debian GNU/Linux 1.3 war nur für `i386` verfügbar. Debian 2.0 wurde für die Architekturen `i386` und `m68k` ausgegeben. Debian 2.1 kam für die Architekturen `i386`, `m68k`, `alpha` und `sparc` heraus. Seither ist Debian enorm gewachsen. Debian 9 unterstützt im Ganzen zehn Linux-Architekturen (`amd64`, `arm64`, `armel`, `armhf`, `i386`, `mips`, `mips64el`, `mipsel`, `ppc64el` und `s390x`) sowie zwei kFreeBSD-Architekturen (`kfreebsd-i386` und `kfreebsd-amd64`).

Informationen für Entwickler und Anwender über die spezifischen Portierung sind auf der Debian-Webseite Portierungen verfügbar.

4.6.3 Pakete

Es gibt zwei Typen von Debian-Paketen, nämlich `source`- und `binary`-Pakete.

Abhängig vom Format des Quellpakets wird es aus einem oder mehreren Dateien zusätzlich zur zwingend notwendigen `.dsc`-Datei bestehen:

- Mit Format "1.0" hat es entweder eine `.tar.gz`-Datei oder sowohl eine `.orig.tar.gz`- als auch eine `.diff.gz`-Datei;
- Mit Format "3.0" (quilt), hat es zwingend eine Tarball-Archivdatei von Upstream mit der Endung `.orig.tar.{gz,bz2,xz}`, optional mehrere `.orig-`*Komponente*`.tar.{gz,bz2,xz}`-Debian-Tarball-Archive;
- Mit Format "3.0" (nativ) hat es nur eine einzelne `.tar.{gz,bz2,xz}`-Tarball-Archivdatei.

Falls das Paket speziell für Debian entwickelt wurde und nicht außerhalb von Debian verteilt wird, gibt es dort nur eine `.tar.{gz,bz2,xz}`-Datei, die den Quellcode des Programms enthält, welches dann "natives" Quellpaket genannt wird. Falls ein Paket auch anderswo verteilt wird, wird in der `.orig.tar.{gz,bz2,xz}`-Datei der sogenannte `Upstream Source Code` gespeichert, das ist der Quellcode vom `Upstream Maintainer` (oft dem Autor der Software). In diesem Fall enthalten die Dateien `.diff.gz` oder `debian.tar.{gz,bz2,xz}` die Änderungen, die durch den Debian-Betreuer vorgenommen wurden.

Die `.dsc`-Datei listet alle Dateien im Quellpaket auf, zusammen mit den Prüfsummen (`md5sums`, `sha1sums`, `sha256sums`) und einigen zusätzlichen Informationen über das Paket (Betreuer, Version etc.).

4.6.4 Distributionen

Das im vorherigen Kapitel beschriebene Verzeichnissystem ist selbst innerhalb der `distribution directories` enthalten. Jede Distribution ist letztlich im Verzeichnis `pool` in der obersten Ebene des Debian-Archivs selbst enthalten.

Zusammenfassend gesagt hat das Debian-Archiv ein Wurzelverzeichnis auf einem FTP-Server. Auf der Spiegel-Seite `ftp.us.debian.org` ist beispielsweise das Debian-Archiv selbst in /debian enthalten, was ein gebräuchlicher Ort dafür ist (ein anderer ist `/pub/debian`).

Eine Distribution umfasst Debian-Quell- und -Binärpakete und die jeweiligen Indexdateien `Sources` sowie `Packages`, die die Kopfzeileninformationen von all diesen Paketen enthalten. Erstere werden im Verzeichnis `pool/` aufbewahrt, während letztere im Verzeichnis `dists/` des Archivs liegen (aus Gründen der Rückwärtskompatibilität).

4.6.4.1 Stable, testing und unstable

Es gibt immer die Distributionen mit den Namen `Stable` (angesiedelt in `dists/stable`), `Testing` (angesiedelt in `dists/testing`) und `Unstable` (angesiedelt in `dists/unstable`). Dies spiegelt den Entwicklungsprozess des Debian-Projekts wider.

In der Distribution `Unstable` wird aktiv entwickelt (daher wird diese Distribution manchmal auch die `development distribution` genannt). Jeder Debian-Entwickler kann jederzeit seine Pakete in dieser Distribution ändern. Daher ändert sich der Inhalt dieser Distribution von Tag zu Tag. Da keine besonderen Anstrengungen unternommen werden, um sicherzustellen, dass alles in dieser Distribution funktioniert, ist sie manchmal buchstäblich instabil.

Die Distribution *Testing* wird automatisch aus Paketen von `Unstable` erzeugt, falls sie bestimmte Voraussetzungen erfüllen. Diese Voraussetzungen sollten eine gute Qualität für Pakete innerhalb von `Testing` gewährleisten. Die Aktualisierung von `Testing` wird zweimal täglich lanciert, gleich nachdem die neuen Pakete installiert wurden. Siehe *Die Distribution Testing*.

Nach einer Entwicklungsperiode, sobald die Releasemanager diese als abgeschlossen erachten, wird die Distribution `Testing` eingefroren. Das bedeutet, dass die Richtlinien, die steuern, welche Pakete von `Unstable` nach `Testing` verschoben werden, verschärft werden. Pakete, die zu viele Fehler aufweisen, werden entfernt. In `Testing` sind außer Fehlerkorrekturen keine anderen Änderungen erlaubt. Nachdem, abhängig vom Fortschritt, einige Zeit verstrichen ist, wird die Distribution `Testing` sogar noch weiter eingefroren. Einzelheiten darüber, wie die Distribution `Testing` gehandhabt wird, werden vom Veröffentlichungs-Team auf "debian-devel-announce" publiziert. Nachdem die offenen Probleme zur Zufriedenheit des Veröffentlichungs-Teams gelöst wurden, wird die Distribtion veröffentlicht. Veröffentlichen bedeutet, dass `Testing` in `Stable` umbenannt wird und für das neue `Testing` eine neue Kopie erstellt wird. Das vorherige `Stable` wird in `Oldstable` umbenannt und bleibt bis zur endgültigen Archivierung dort. Bei der Archivierung wird der Inhalt nach `archive.debian.org` verschoben.

Der Entwicklungszyklus hängt von der Annahme ab, dass die Distribution `Unstable` nach einer Periode, in der sie `Testing` durchläuft, `stabil` geworden ist. Sogar wenn eine Distribution stabil geworden ist, verbleiben zwangsläufig ein paar Fehler – daher wird die stabile Distribution ab und zu aktualisiert. Diese Aktualisierungen wurden je-

doch sehr gründlich getestet und werden in einem einzelnen Archiv eingeführt, um das Risiko zu vermindern, neue Fehler einzuschleppen. Sie können die geplanten Ergänzungen zu `Stable` im Verzeichnis `proposed-updates` finden. Diese Pakete in `proposed-updates`, die den Anforderungen entsprechen, werden periodisch in einem Stapellauf in die Stable-Distribution verschoben und die Überarbeitungsstufe der Stable-Distribution wird erhöht (z.B. "6.0" wird "6.0.1", "5.0.7" wird "5.0.8" und so weiter). Bitte sehen Sie unter *Sonderfall Uploads in die Distributionen Stable und Oldstable* nach, um weitere Einzelheiten zu erfahren.

Beachten Sie, dass die Entwicklung in `unstable` während des Freeze nicht wie gewohnt fortgesetzt werden sollte, da Pakete immer noch in `unstable` erstellt werden, bevor diese nach `testing` migriert werden. Daher sollte `unstable` nur Pakete enthalten die auch nach `testing` migrieren können müssen. Laden Sie daher Pakete nur während des Freeze nach `unstable` hoch, wenn Sie einen Unblock-Request anfordern möchten (oder wenn sich das Paket nicht in `testing` befindet).

Wenn Sie an neuen Dingen für die Zeit nach dem Freeze arbeiten laden Sie stattdessen Pakete besser nach `experimental`.

4.6.4.2 Weitere Informationen über die Testing-Distribution

Pakete werden üblicherweise in der Distribution `Testing` installiert, nachdem sie in `Unstable` gewissen Tests unterzogen wurden.

Lesen Sie bitte die *Die Distribution Testing*, um weitere Einzelheiten zu erfahren.

4.6.4.3 Experimental

Die Distribution `Experimental` ist eine Spezialdistribution. Sie ist keine vollständige Distribution im Sinn von `Stable`, `Testing` und `Unstable`. Stattdessen ist sie als temporärer Sammelpunkt für hoch experimentelle Software gedacht, bei der eine gute Chance besteht, das ganze System zu zerstören oder von Software, die sogar für die Distribution `Unstable` noch zu instabil ist (bei der es aber dennoch einen Grund gibt, sie zu paketieren). Bei Benutzern, die Pakete aus `Experimental` herunterladen und installieren, wird davon ausgegangen, dass sie ausreichend gewarnt wurden. Kurz gesagt, ist in der Distribution `Experimental` alles möglich.

Dies sind die Zeilen der sources.list 5 für `Experimental`:

```
deb http://deb.debian.org/debian/ experimental main
deb-src http://deb.debian.org/debian/ experimental main
```

Falls eine Chance besteht, dass die Software ein System schwer beschädigen könnte, ist es wahrscheinlich besser, sie in `Experimental` abzulegen. Zum Beispiel sollte ein experimentell komprimiertes Dateisystem wahrscheinlich nach `Experimental` wandern.

Jedes Mal wenn eine neue Originalversion eines Pakets vorliegt, das neue Funktionen mitbringt, aber ältere beschädigt, sollte es entweder nicht oder nach `Experimental` hochgeladen werden. Eine neue Beta-Version einer Software, die eine völlig unterschiedliche Konfiguration nutzt, kann nach Ermessen des Betreuers nach `Experimental` wandern. Falls Sie an einer inkompatiblen oder komplexen Situation bei Upgrades arbeiten, können sie `Experimental` auch als Sammelpunkt benutzen, damit Tester frühzeitig Zugriff darauf haben.

Trotzdem kann experimentelle Software unter machen Umständen, mit ein paar Warnungen in der Beschreibung, bewusst nach `Unstable` wandern, dies wird aber nicht empfohlen, da von Paketen aus `Unstable` erwartet wird, dass diese nach `Testing` und final dann nach `Stable` wandern kann. Sie sollten keine Angst haben `Experimental` zu benutzen, da es dem FTP-Master keine Probleme bereitet. Die experimentellen Pakete werden regelmäßig entfernt, wenn Sie ein Paket mit einer höheren Versionsnummer nach `Unstable` hochladen.

Neue Software, die das System voraussichtlich nicht beschädigt, kann direkt nach `Unstable` hochgeladen werden.

Eine Alternative zu `Experimental` ist die Benutzung Ihres persönlichen Webspaces auf `people.debian.org`.

4.6.5 Codenamen der Veröffentlichungen

Jede veröffentlichte Debian-Distribution hat einen `Codenamen`: Debian 11 wurde `bullseye` genannt, Debian 12 `bookworm`, Debian 13 `trixie`, die nächste Debian-Veröffentlichung 14 wird `forky` heißen und Debian 15 wird den Namen `duke` erhalten. Außerdem gibt es eine Pseudodistribution namens `Sid`, die die derzeitige `Unstable`-Distribution ist. Da Pakete von `Unstable` nach `Testing` wandern, wenn Sie stabiler werden, wird `Sid` selbst nie

veröffentlicht. Neben dem üblichen Inhalt der Debian-Distribution enthält `Sid` auch Pakete für Architekturen, die noch nicht offiziell durch Debian unterstützt oder veröffentlicht werden. Es ist vorgesehen, diese Architekturen an irgendeinem zukünftigen Datum in die Hauptdistribution zu integrieren. Die Codenamen und Versionen älterer Veröffentlichungen sind auf dieser Webseite aufgeführt.

Da Debian ein offenes Entwicklungsmodell hat (d.h. jeder kann teilnehmen und der Entwicklung folgen), werden sogar die Distributionen `Unstable` und `Testing` über das Internet durch die Debian-FTP- und -HTTP-Netzwerkserver verteilt. Folglich müsste, falls das Verzeichnis, das die Kandidatenversion für die Veröffentlichung enthält, `testing` genannt würde, es beim Veröffentlichen der Version in `stable` umbenannt werden, was dazu führen würde, dass alle FTP-Spiegel die ganze Distribution erneut erhalten müssten (die ziemlich groß ist).

Wären andererseits von Anfang an die Distributionsverzeichnisse `Debian-x.y` genannt worden, würden die Leute denken, die Debian-Veröffentlichung `x.y` sei verfügbar. (Dies ist früher einmal passiert, ein CD-Verkäufer erstellte eine Debian 1.0 CD-ROM auf Bais einer pre-1.0-Entwicklerversion. Das ist der Grund, weshalb die erste offizielle Debian-Veröffentlichung 1.1 und nicht 1.0 war.)

Dadurch werden die Namen der Distributionsverzeichnisse im Archiv durch ihre Codenamen festgelegt und nicht durch ihren Veröffentlichungsstatus (z.B. "`trixie`"). Diese Namen bleiben während der Entwicklungszeit und nach der Veröffentlichung erhalten. Symbolische Verweise, die einfach geändert werden können, geben die aktuell veröffentlichte stabile Distribution an. Das ist der Grund, weshalb die echten Distributionsverzeichnisse `Codenamen` benutzen, während symbolische Verweise für `stable`, `testing` und `unstable` auf die entsprechenden Veröffentlichungsverzeichnisse verweisen.

4.7 Debian-Spiegel

Für die verschiedenen Archive zum Herunterladen und die Website stehen mehrere Spiegel zur Verfügung, um die regulären Server vor zu großer Auslastung zu bewahren. Eigentlich sind einige der regulären Server nicht öffentlich zugänglich – eine erste Schicht von Spiegeln balanciert stattdessen die Last aus. Auf diese Art greifen die Anwender immer auf die Spiegel zu und sind es gewohnt, sie zu benutzen. Dies ermöglich es Debian, seine Bandbreitenanforderungen besser über mehrere Server und Netzwerke zu verteilen und vermeidet grundsätzlich, dass Benutzer übermäßig einen primären Server belasten. Beachten Sie, dass die erste Schicht von Spiegelservern so aktuell wie möglich ist, da ihre Aktualisierung durch die internen Sites ausgelöst wird (dies wird "push mirroring" genannt).

All die Informationen über Debian-Spiegel, einschließlich der verfügbaren FTP-/HTTP-Server, sind unter https://www.debian.org/mirror/ zu finden. Diese nützliche Seite enthält auch Informationen und Werkzeuge, die hilfreich sein können, falls Sie daran interessiert sind, entweder für internen oder öffentlichen Zugriff Ihren eigenen Spiegel einzurichten.

Beachten Sie, dass Spiegel generell durch Dritte betrieben werden, die ein Interesse daran haben, Debian zu helfen. Daher haben Entwickler generell keine Konten auf diesen Maschinen.

4.8 Das Incoming-System

Das Incoming-System ist dafür verantwortlich, aktualisierte Pakete zu sammeln und im Debian-Archiv zu installieren. Es besteht aus einer Zusammenstellung von Verzeichnissen und Skripten, die auf `ftp-master.debian.org` installiert sind.

Pakete werden von den Betreuern in ein Verzeichnis hochgeladen, das `UploadQueue` heißt. Dieses Verzeichnis wird alle paar Minuten durch einen Daemon gescannt, der `queued` genannt wird; es werden `*.command`-Dateien ausgeführt und verbleibende, korrekt signierte `*.changes`-Dateien werden zusammen mit ihren zugehörigen Dateien in das Verzeichnis `unchecked` verschoben. Dieses Verzeichnis ist für die meisten Entwickler unsichtbar, da FTP-Master eingeschränkt wurde. Es wird alle 15 Minuten durch das Skript `dak process-upload` gelesen, das die Richtigkeit der hochgeladenen Pakete und ihre kryptografischen Signaturen prüft. Falls das Paket für installationsbereit befunden wird, wird es in das Verzeichnis `done` verschoben. Falls dies das erste Hochladen des Pakets ist (oder es neue Binärpakete enthält), wird es in das Verzeichnis `new` verschoben, wo es auf die Freigabe durch die FTP-Master wartet. Falls das Paket Dateien enthält, die manuell installiert werden müssen, wird es in das Verzeichnis `byhand` verschoben, wo es auf die manuelle Installation durch die FTP-Master wartet. Andernfalls, falls ein Fehler aufgespürt wurde, wird das Paket abgewiesen und landet im Verzeichnis `reject`.

Sobald das Paket akzeptiert wurde, sendet das System dem Betreuer per E-Mail eine Bestätigung, schließt alle Fehler, die durch das Hochladen als behoben markiert wurden und die Auto-Builder können beginnen, es erneut zu kompilieren. Das Paket ist nun öffentlich unter https://incoming.debian.org/ zugänglich, bis es wirklich im Debian-Archiv installiert wird. Dies geschieht viermal täglich (und wird aus historischen Gründen "dinstall run" genannt). Das Paket wird dann aus dem Incoming-System entfernt und zusammen mit den anderen Paketen in den Pool installiert. Sobald alle anderen Aktualisierungen (Erzeugen neuer `Packages`- und `Sources`-Indexdateien zum Beispiel) durchgeführt wurden, wird ein Spezialskript aufgerufen, das alle Primärspiegel auffordert, sich selbst zu aktualisieren.

Die Archivverwaltungs-Software wird außerdem die von Ihnen hochgeladene OpenPGP signierte `.changes`-Datei an die entsprechenden Mailinglisten senden. Falls ein Paket veröffentlicht wird, bei dem die Distribution auf `stable` gesetzt ist, wird die Ankündigung an `debian-changes@lists.debian.org` gesandt. Falls ein Paket veröffentlicht wird, bei dem die Distribution auf `unstable` oder `experimental` gesetzt ist, wird die Ankündigung stattdessen an `debian-devel-changes@lists.debian.org` oder an `debian-experimental-changes@lists.debian.org` gesandt.

Obwohl der Zugriff auf FTP-Master eingeschränkt ist, ist eine Kopie für alle Entwickler unter `mirror.ftp-master.debian.org` verfügbar.

4.9 Paketinformationen

4.9.1 Im Web

Jedes Paket hat mehrere zugehörige Web-Seiten. `https://packages.debian.org/`*Paketname* zeigt jede Version des Pakets, die in verschiedenen Distributionen verfügbar ist. Jede Version verweist auf eine Seite mit Informationen, einschließlich der Paketbeschreibung, der Abhängigkeiten und der Verweise zum Herunterladen.

Die Fehlerdatenbank verfolgt Fehler für jedes Paket. Sie können die Fehler unter der URL `https://bugs.debian.org/`*Paketname* ansehen.

4.9.2 Das Hilfswerkzeug `dak ls`

`dak ls` ist Teil der Werkzeugsammlung Dak, die verfügbare Paketversionen für alle bekannten Distributionen und Architekturen auflistet. Das Werkzeug `dak` ist auf `ftp-master.debian.org` und auf dem Spiegel `mirror.ftp-master.debian.org` verfügbar. Es benutzt ein einziges Argument, das einem Paketnamen entspricht. Ein Beispiel erklärt es besser:

```
$ dak ls evince
evince     | 3.22.1-3+deb11u2 | oldstable           | source, amd64, arm64, armel,␣
↪armhf, i386, mips, mips64el, mipsel, ppc64el, s390x
evince     | 3.22.1-3+deb11u2 | oldstable-debug     | source
evince     | 3.30.2-3+deb12u1 | stable              | source, amd64, arm64, armel,␣
↪armhf, i386, mips, mips64el, mipsel, ppc64el, s390x
evince     | 3.30.2-3+deb12u1 | stable-debug        | source
evince     | 3.38.2-1         | testing             | source, amd64, arm64, armel,␣
↪armhf, i386, mips64el, mipsel, ppc64el, s390x
evince     | 3.38.2-1         | unstable            | source, amd64, arm64, armel,␣
↪armhf, i386, mips64el, mipsel, ppc64el, s390x
evince     | 3.38.2-1         | unstable-debug      | source
evince     | 40.4-1           | buildd-experimental | source, amd64, arm64, armel,␣
↪armhf, i386, mips64el, mipsel, ppc64el, s390x
evince     | 40.4-1           | experimental        | source, amd64, arm64, armel,␣
↪armhf, i386, mips64el, mipsel, ppc64el, s390x
evince     | 40.4-1           | experimental-debug  | source
```

An diesem Beispiel können Sie sehen, dass sich die Version in `Unstable` von der Version in `Testing` unterscheidet und dass dort ein rein binärer NMU des Pakets für alle Architekturen durchgeführt wurde. Jede Version des Pakets wurde auf allen Architekturen neu kompiliert.

You do not need an account on `ftp-master.debian.org` to get this information. The `rmadison` command, from the `devscripts` package (see *devscripts*), queries the same data over HTTPS; for example, `rmadison --suite=unstable --architecture=amd64 evince` restricts the output to one suite and one architecture. The data is also available directly from the web at `https://api.ftp-master.debian.org/madison?package=evince`.

4.10 Das Debian-Paketverfolgungssystem

Das Debian-Paketverfolgungssystem ist ein E-Mail- und Web-basiertes Werkzeug, das die Aktivität eines Quellpakets verfolgt. Sie können die gleichen E-Mails wie der Paketbetreuer erhalten, indem Sie einfach im Debian-Paketverfolgungssystem das Paket *abonnieren*.

Das Paketverfolgungssystem hat eine Web-Schnittstelle unter https://tracker.debian.org/, die eine große Menge Informationen über jedes Quellpaket zusammenträgt. Sie zeichnet sich durch viele nützliche Verweise aus (BTS, QA-Statistiken, Kontaktinformationen, DDTP-Übersetzungsstatus, Paketerstellungsprotokolle) und sammelt noch viele weitere Informationen von verschiedenen Stellen (die 30 letzten Einträge des Änderungsprotokolls, Teststatus etc.). Es ist ein sehr nützliches Werkzeug, falls Sie wissen möchten, was bei einem speziellen Quellpaket vorgeht. Außerdem können Sie, sobald Sie sich authentifiziert haben, jedes Paket mit einem Klick abonnieren oder abbestellen.

Sie können mit einer URL wie `https://tracker.debian.org/pkg/`*Quellpaket* direkt zur Web-Seite springen, die eine spezielles Quellpaket betrifft.

Für genauere Informationen sollten Sie die Dokumentation lesen. Unter anderem wird dort erklärt, wie Sie mit ihm per E-Mail interagieren, von ihm weitergeleitete E-Mails filtern, Änderungsbenachrichtigungen vom VCS-System konfigurieren oder Features für Paketverwalter und Teams bestmöglichst ausnutzen können.

4.11 Paketübersicht des Entwicklers

Ein Webportal der QS (Qualitätssicherung) ist unter https://qa.debian.org/developer.php verfügbar. Es zeigt eine tabellarische Auflistung aller Pakete eines einzelnen Entwicklers an (einschließlich derer, bei denen eine Gemeinschaft als Mitbetreuer aufgelistet ist). Die Tabelle gibt einen guten Überblick über die Pakete des Betreuers: Anzahl der Fehler nach Schweregrad, Liste der verfügbaren Versionen in jeder Distribution, Test-Status und vieles mehr, einschließlich Verweisen zu irgendwelchen anderen nützlichen Informationen.

Es ist ein guter Tipp, regelmäßig auf seine eigenen Daten zu schauen, um keine offenen Fehler oder Zuständigkeiten für Pakete zu vergessen.

4.12 Debians FusionForge-Installation: Alioth

Alioth war ein Debian Dienst der im Juni 2018 abgelöst und danach abgeschaltet wurde. Es basierte auf einer leicht modifizierten Version der FusionForge-Software (die aus SourceForge und GForge entwickelt wurde). Diese Software bot Entwicklern Zugriff auf benutzerfreundliche Tools wie Bug-Tracker, Patch-Manager, Projekt-/Task-Manager, Datei-Hosting-Dienste, Mailinglisten, VCS-Repositorys usw.

Für viele zuvor angebotene Dienste gibt es Ersatz. Dies ist wichtig zu wissen, da es immer noch viele Hinweise auf Alioth gibt, die noch korrigiert werden müssen. Wenn Sie auf solche Referenzen stoßen, nehmen Sie sich bitte die Zeit, um zu versuchen, diese anzupassen, indem Sie beispielsweise Fehler melden oder die Referenz nach Möglichkeit selbst korrigieren.

4.13 Goodies für Debian Mitglieder

Vorteile für Debian Mitglieder sind unter https://wiki.debian.org/MemberBenefits dokumentiert.

KAPITEL 5

Pakete verwalten

Dieses Kapitel enthält Informationen, die sich auf das Erstellen, Hochladen, Verwalten und Portieren von Paketen beziehen.

5.1 Neue Pakete

Falls Sie ein neues Paket für die Debian-Distribution erstellen möchten, sollten Sie zuerst die Liste der Arbeitbedürfenden und voraussichtlichen Pakete (WNPP) prüfen. Die Prüfung der WNPP-Liste stellt sicher, dass nicht bereits jemand an der Paketierung dieser Software arbeitet und kein doppelter Aufwand betrieben wird. Weitere Informationen finden Sie auf den WNPP-Web-Seiten.

Ausgehend von der Annahme, dass noch niemand an Ihrem zukünftigen Paket arbeitet, müssen Sie einen Fehlerbericht (*Fehler berichten*) gegen das Pseudopaket `wnpp` einreichen, in dem Sie Ihr Vorhaben vorstellen. Der Fehlerbericht muss mindestens eine Beschreibung des neuen Pakets enthalten (sodass andere es überprüfen können), die Lizenz angeben die dem Paket zugedacht werden soll, sowie die derzeitige URL, von der es heruntergeladen werden kann.

Sie sollten den Betreff des Fehlers auf `ITP:` *Paketname* `--` *kurze Beschreibung* setzen, wobei Sie *Paketname* durch *den Namen Ihres Pakets* ersetzen. Der Schweregrad des Fehlerberichts muss auf `wishlist` gesetzt werden. Bitte senden Sie mit der Kopfzeile X-Debbugs-CC eine Kopie an `debian-devel@lists.debian.org` (benutzen Sie nicht CC:, da in diesem Fall der Betreff der Nachricht die Fehlernummer nicht angibt). Falls Sie so viele (mehr als zehn) neue Pakete paketieren, dass die Benachrichtigung auf der Liste als störend empfunden würde, senden Sie stattdessen nach dem Einreichen des Fehlers eine Zusammenfassung an die Liste "debian-devel". Dies wird andere Entwickler über die bevorstehenden Pakete informieren und eine Überprüfung Ihrer Beschreibung und Ihres Paketnamens ermöglichen.

Bitte fügen Sie im Änderungsprotokoll des neuen Pakets den Eintrag `Closes: #`*nnnnn* hinzu, um den Fehlerbericht automatisch zu schließen, sobald das neue Paket im Archiv installiert wird (siehe *Wann Fehler durch neue Uploads geschlossen werden*).

Falls Sie der Ansicht sind Ihr Paket bedürfe einiger Erklärungen für die Administratoren der Paketwarteschlange NEW, so fügen Sie diese dem Änderungsprotokoll hinzu, senden Sie die E-Mail die Sie als Betreuer nach dem Upload zurückbekommen haben an `ftpmaster@debian.org` oder antworten Sie auf die ablehnende E-Mail für den Fall, dass Sie bereits erneut hochladen.

Wenn sicherheitskritische Fehler geschlossen werden, fügen Sie die CVE-Nummern sowie `Closes: #`*nnnnn* bei. Dies ist nützlich zur Verfolgung von Schwachstellen durch das Sicherheits-Team. Falls etwas hochgeladen wird, bevor die ID der Sicherheitsankündigung bekannt ist, sollte beim nächsten Upload der historische Änderungspro-

tokolleintrag geändert werden. Bitte fügen Sie sogar in diesem Fall dem Original-Änderungsprotokolleintrag alle verfügbaren Hinweise auf Hintergrundinformationen hinzu.

Es gibt eine Vielzahl von Gründen weshalb Paketbetreuer um die Ankündigung ihrer Absichten gebeten werden:

- Es hilft dem (möglicherweise neuen) Betreuer, die Erfahrung der Leute auf der Liste anzuzapfen und teilt ihm mit, ob jemand anderes bereits daran arbeitet.
- Es zeigt anderen Leuten, die darüber nachdenken am Paket zu arbeiten, dass es bereits einen Freiwilligen gibt, so dass der Aufwand verteilt werden kann.
- Es sagt den übrigen Betreuern mehr über das Paket, als nur aus Beschreibungszeile erkennbar und der übliche Eintrag im Änderungsprotokoll "Initial release", der an `debian-devel-changes@lists.debian.org` gesandt wird.
- Es ist hilfreich für Leute, die die Veröffentlichung `Unstable` für die tägliche Arbeit benutzen (und die die vorderste Frontlinie von Testern bilden). Diese Leute sollten ermutigt werden.
- Die Ankündigungen geben Betreuern und anderen interessierten Parteien ein besseres Gefühl dafür, was gerade geschieht und was im Projekt neu ist.

5.2 Änderungen im Paket aufzeichnen

Von Ihnen vorgenommene Änderungen müssen in `debian/changelog` in einer Form aufgezeichnet werden, so dass andere Personen die Änderungen nachvollziehen uns verstehen können. Diese Änderungen sollten eine kurzgefasste Beschreibung bereitstellen, was geändert wurde, warum (falls dies zweifelhaft ist) und vermerken, ob irgendwelche Fehler geschlossen wurden. Sie zeichnen außerdem auf, wenn das Paket vervollständigt wurde. Diese Datei wird in `/usr/share/doc/`*Paket*`/changelog.Debian.gz` (oder `/usr/share/doc/`*Paket*`/changelog.gz` für native Pakete) installiert.

Die Datei `debian/changelog` hat eine bestimmte Struktur mit einer Anzahl unterschiedlicher Felder. Ein bedeutendes Feld, die `distribution`, wird in *Eine Distribution auswählen* beschrieben. Weitere Informationen über die Struktur dieser Datei sind im Abschnitt `debian/changelog` der Debian-Richtlinien zu finden.

Wenn das Paket im Archiv installiert ist, können Einträge im Änderungsprotokoll benutzt werden, um Debian-Fehler automatisch zu schließen. Siehe *Wann Fehler durch neue Uploads geschlossen werden*.

Es ist üblich, dass der Änderungsprotokolleintrag eines Pakets, der eine neue Originalversion der Software enthält, wie folgt aussieht:

```
* New upstream release.
```

There are tools to help you create entries and finalize the `changelog` for release — see *devscripts* (command `dch`), `git-buildpackage` (command `gbp dch`) and *elpa-dpkg-dev-el*.

Siehe auch *Optimale Vorgehensweisen für debian/changelog*.

5.3 Das Paket testen

Bevor Sie Ihr Paket hochladen, sollten Sie es grundlegenden Tests unterziehen. Zumindest sollten Sie die folgenden Dinge ausprobieren (Sie sollten eine ältere Version des gleichen Debian-Pakets zur Hand haben):

- Führen Sie für das Paket `lintian` aus. Sie können `lintian` wie folgt ausführen: `lintian -v`*Paket-Version*`.changes`. Falls Sie die von `lintian` erzeugte Ausgabe nicht verstehen, versuchen Sie den Schalter `-i` hinzuzufügen. Er veranlasst `lintian` eine viel detailliertere Beschreibung des Problems auszugeben.

  Normalerweise sollte ein Paket *nicht* hochgeladen werden, wenn es `lintian`-Fehler (diese beginnen mit `E`) verursacht.

  Weitere Informationen über `lintian` finden Sie unter *lintian*.

- Führen Sie wahlweise `debdiff` (siehe *debdiff*) aus, um Änderungen von einer älteren Version zu analysieren, falls es eine solche gibt.

- Installieren Sie das Paket und prüfen die korrekte Funktion dessen in einem tagesaktuellem `Unstable` System.
- Aktualisieren Sie das Paket von einer vorherigen Version auf Ihre neu erstellte Version.
- Entfernen Sie das Paket und installieren Sie es erneut.
- Die Installation, eine Aktualisierung oder auch ein komplettes Entfernen des Paketes können manuell oder auch durch das `piuparts` Werkzeug getestet werden.
- Führen Sie nach der Installation des Pakets `adequate <pakage name>` aus um zu prüfen, dass der installierte Zustand keine Richtlinien verletzt. Idealerweise sollte dies mit einem `autopkgtest` (siehe *adequate*) automatisiert werden.
- Kopieren Sie das Quellpaket in ein anderes Verzeichnis und versuchen Sie es zu entpacken und neu zu erstellen. Dies testet, ob das Paket bestehende Dateien von außerhalb des Tarballs benötigt oder ob es auf Benutzerrechten beruht, die in den Dateien konserviert sind, die innerhalb der `.diff.gz`-Datei mitgeliefert wurden.

5.4 Layout des Quellpakets

Es gibt zwei Typen von Debian-Quellpaketen:

- Die sogenannten `native`-Pakete, bei denen es keine Unterschiede zwischen dem Originalquellcode und den auf Debian angewandten Patches gibt.
- Die (häufigeren) Pakete, bei denen der originale Quellcode-Tarball mit einer anderen Datei mitgeliefert wird, die die von Debian vorgenommenen Änderungen enthält.

Bei nativen Paketen enthält der Quell-Tarball eine Steuerungsdatei für Debian-Quellen (`.dsc`) und einen Quell-Tarball (`.tar.{gz,bz2,xz}`). Ein Quellpaket eines nicht nativen Paketes enthält eine Steuerungsdatei für Debian-Quellen, den Original-Quellcode-Tarball (`.orig.tar.{gz,bz2,xz}`) und die Debian-Änderungen (`.diff.gz` für das Quellformat "1.0" oder `.debian.tar.{gz,bz2,xz}` für das Quellformat "3.0 (quilt)").

Mit dem Quellformat "1.0" wurde zur Zeit des Erstellens durch `dpkg-source` festgelegt, ob ein Paket nativ ist oder nicht. Heutzutage wird empfohlen, das gewünschte Quellformat explizit durch Angabe von "3.0 (quilt)" oder "3.0 (native)" in `debian/source/format` festzulegen. Der Rest dieses Abschnitts bezieht sich auf nicht native Pakete.

Anfangs, wenn eine Version hochgeladen wird, die einer bestimmten Version des Ursprungsquellcodes entspricht, muss die Original-Tar-Quelldatei hochgeladen und in die `.changes`-Datei eingefügt werden. Nachfolgend sollte eben diese Tar-Datei benutzt werden, um neue .diff- und `.dsc`-Dateien zu erstellen ohne die Notwendigkeit, diese erneut hochzuladen.

Standardmäßig werden `dpkg-genchanges` und `dpkg-buildpackage` die Original-Tar-Quelldatei nur dann einbeziehen, falls der aktuelle Änderungsprotokolleintrag eine andere Originalversion des vorhergehenden Eintrags hat. Dieses Verhalten kann durch die Benutzung der Option `-sa` geändert werden, um es immer einzubeziehen oder `-sd`, um es immer wegzulassen.

Falls im Upload kein Originalquellcode enthalten ist, *muss* die Original-Tar-Quelldatei, die von `dpkg-source` benutzt wurde, um die `.dsc`-Datei und die hochzuladene Diff-Datei zu erzeugen, Byte für Byte identisch mit der sein, die bereits im Archiv ist.

Bitte beachten Sie, dass in nicht nativen Paketen Zugriffsrechte von Dateien, die nicht in den `*.orig.tar.{gz,bz2,xz}`-Dateien enthalten sind, nicht aufbewahrt werden, da das Diff die Dateizugriffsrechte nicht im Patch speichert. Wenn Sie jedoch das Format "3.0 (quilt)" benutzen, werden Zugriffsrechte von Dateien innerhalb des `debian`-Verzeichnisses aufbewahrt, da sie in einem Tar-Archiv gespeichert werden.

5.5 Eine Distribution auswählen

Bei jedem Upload muss angegeben werden, für welche Distribution das Paket gedacht ist. Der Prozess der Paketerstellung extrahiert diese Information aus der ersten Zeile der Datei `debian/changelog` und platziert sie im Feld `Distribution` der `.changes`-Datei.

Normalerweise werden Pakete nach `Unstable` hochgeladen. Beim Hochladen nach `Unstable` oder `Experimental` sollte diese Suite-Namen im Änderungsprotokolleintrag verwendet werden. Beim Hochladen für andere unterstützte Suites sollten die Suite-Codenamen benutzt werden, um Mehrdeutigkeiten zu vermeiden.

Außerdem gibt es auch noch andere mögliche Distributionen: *codename*`-security`, aber lesen Sie *Handhabung von sicherheitsrelevanten Fehlern*, um weitere Informationen darüber zu erhalten.

Es ist nicht möglich ein Paket gleichzeitig in mehrere Distributionen hochzuladen.

5.5.1 Sonderfall Uploads in die Distributionen `Stable` und `Oldstable`

Hochladen nach `Stable` bedeutet, dass das Paket in die Warteschlange `proposed-updates-new` übertragen wird, damit es von den Veröffentlichungsverwaltern überprüft werden kann. Falls es zugelassen wird, wird es in das Verzeichnis `stable-proposed-updates` des Debian-Archivs installiert. Von dort wird es zum nächsten Veröffentlichungszeitpunkt in `Stable` eingefügt.

Uploads in eine unterstützte `Stable` Version sollten den Suite-Namen im Änderungslog enthalten, also zum Beispiel `trixie` oder `bookworm`. Üblicherweise sollten Sie `reportbug` benutzen um einen Bugreport gegen das Pseudopaket `release.debian.org` zu erstellen, und in diesem eine Ausgabe von `debdiff` (basierend von der aktuellen Paketversion) gegen die momentan aktuelle Version in `Stable` anhängen wenn Sie ein Paket in einer der aktuellen `Stable` Veröffentlichungen aktualisieren wollen. Ebenso sollten Sie innerhalb dieses Bugreports begründen warum Sie die Version in `Stable` aktualisieren wollen. Sofern es zugehörige andere Fehlernummern gibt, die in Bezug zur vorbereiteten neuen Version für `Stable` stehen, geben Sie diese mit an. Die Release-Manager entscheiden über das weitere Vorgehen und werden Ihnen weitere Information zukommen lassen.

Wenn Sie sicher sind, dass der Upload ohne Änderungen akzeptiert werden wird, können Sie ihn gleichzeitig mit dem Einreichen des Fehlerreports gegen `release.debian.org` hochladen. Wenn Sie mit dem Prozess noch nicht vertraut sind, empfehlen wir Ihnen, vor dem Hochladen eine Genehmigung einzuholen, damit Sie sehen können, ob Ihre Erwartungen mit denen des Veröffentlichungsteam übereinstimmen.

Either way, there must be an accompanying bug for tracking, and your upload must comply with these acceptance criteria defined by the stable release managers. These criteria are designed to help the process be as smooth and frustration-free as possible.

- Der Fehler, den Sie in `Stable` beheben möchten, muss bereits in `Unstable` behoben sein (und das Paket mit der korrigierten Version darf nicht in NEW oder der Delayed-Queue liegen).
- Der Fehlerreport muss die Dringlichkeit "Important" oder höher besitzen.
- Die Metadaten des Fehlerreports - im Speziellen die betroffenen Versionen - müssen aktuell sein.
- Fehlerbehebungen müssen so klein als möglich aber auch relevant sein und und einen ausreichend detaillierten Änderungsprotokolleintrag enthalten.
- Ein Quell-Debdiff der vorgeschlagenen Änderung muss in Ihrer Anfrage enthalten sein (nicht nur die nackten Patches oder "Ein Debdiff kann unter $URL gefunden werden").
- Das vorgeschlagene Paket benötigt eine korrekte Versionsnummer (z.B. `...+deb13u1/...~deb13u1` für `trixie` oder `+deb12u1/~deb12u1` für `bookworm`) und Sie sollten in der Lage sein zu erklären welche Tests vorgenommen wurden. Im Debian Richtlinienhandbuch sind Informationen zu der Versionsnummer unter: https://www.debian.org/doc/debian-policy/ch-controlfields.html#special-version-conventions.
- Das Update muss in einer `Stable` Umgebung oder Chroot erstellt worden sein (oder `Oldstable` wenn dies die Zielveröffentlichung ist).
- Korrekturen für Sicherheitsprobleme sollten mit dem Sicherheitsteam abgestimmt werden, es sei denn, Sie haben ausdrücklich angegeben, dass das Sicherheitsteam keinen `DSA` für den Fehler erstellt hat (z. B. über einen "no-dsa"-Marker im *Debian Security Tracker*).
- Sie sollten `release.debian.org` Fehler in debian/changelog nicht selber schließen. Die Fehler werden vom Release Team geschlossen, wenn das Paket die jeweilige Zwischenveröffentlichung erreicht hat.

Es wird empfohlen, `reportbug` zu verwenden, da dies die Erstellung von Fehlern mit korrekten Metadaten erleichtert. Das Veröffentlichungsteam verwendet in großem Umfang Usertags, um Anforderungen zu sortieren und zu verwalten. Falsch gekennzeichnete Berichte können länger dauern, bis sie bemerkt und verarbeitet werden.

Uploads nach Oldstable-Distributionen sind möglich, solange diese nicht archiviert sind. Es gelten die gleichen Regeln wie für Stable.

Früher wurden Uploads nach Stable benutzt, um auch Sicherheitsprobleme anzugehen. Diese Praxis ist jedoch missbilligt, da Uploads für Debian-Sicherheitsankündigungen (DSAs) automatisch in das entsprechende Archiv proposed-updates kopiert werden, wenn die Ankündigung veröffentlicht wird. Weitere detailliertere Informationen über die Handhabung von Sicherheitsproblemen finden Sie unter *Handhabung von sicherheitsrelevanten Fehlern*. Falls das Sicherheits-Team das Problem als zu harmlos erachtet, um es durch ein DSA zu beheben, sind die Veröffentlichungsverwalter normalerweise trotzdem bereit, Ihre Fehlerbehebung bei einem regulären Upload nach Stable einzufügen.

5.5.2 Sonderfall stable-updates Suite

Manchmal entscheiden die Veröffentlichungsmanager, dass ein Update für Stable den Benutzern früher als zur nächsten geplanten Point-Release zur Verfügung gestellt werden soll. In solchen Fällen können Sie das Update in die Suite stable-updates kopieren, die Verwendung dieser Suite wird standardmäßig vom Installationsprogramm aktiviert.

Der Prozess hierzu ist der Gleiche wie in *Sonderfall Uploads in die Distributionen Stable und Oldstable*. Wenn Sie der Meinung sind, dass Ihr Pakte über stable-updates veröffentlicht werden soll, erwähnen Sie dies in Ihrer Anfrage an das Veröffentlichungsteam. Beispiele für Umstände, unter denen sich der Upload für eine solche Behandlung qualifizieren kann, sind:

- Das Update ist dringend jedoch nicht sicherheitsrelevant. Beispiele hierfür sind Pakete, die durch den Zeitfluss unterbrochen wurden (vgl. spamassassin und das Problem des Jahres 2010) und Korrekturen für Fehler, die durch Point-Releases verursacht wurden (z.B. Regressions). Sicherheitsupdates werden weiterhin unabhängig hiervon durch das Sicherheitsarchiv getätigt.
- Das fragliche Paket ist einen Datenpaket und muss zeitnah aktualisiert werden (z.B. tzdata).
- Korrekturen an mehrschichtigen Paketen, die durch externe Änderungen disfunktional wurden (z. B. Tools zum Herunterladen von Videos und tor).
- Pakete, die aktuell sein müssen, um nützlich zu sein (z. B. clamav).
- Uploads nach Stable-updates sollten wie gewohnt den Namen der Suite im Changelog enthalten, z.B. trixie.

Sobald der Upload für proposed-updates akzeptiert wurde und zur Veröffentlichung bereit steht, kopieren die Veröffentlichungsmanager diesen in die Suite stable-updates und geben über die Mailingliste debian-stable-announce ein Stable Update Announcement (SUA) heraus.

Alle Aktualisierungen in stable-updates werden auch in der nächsten Point-Release von Stable enthalten sein.

5.5.3 Sonderfall Uploads nach testing/testing-proposed-updates

Bitte lesen Sie die Informationen im *Direkte Aktualisierungen für Testing*, um weitere Einzelheiten zu erfahren.

5.6 Ein Paket hochladen

5.6.1 Source und Binär Uploads

Jeder Upload in Debian beinhaltet eine signierte Datei .changes in der die angeforderten Änderungen am Archive beschrieben sind, plus ebenfalls das Source und Binärpaket welche durch die Datei .changes referenziert werden.

Wenn möglich sollte die Version des Uploads für das Paket eine source-only Änderungs-Datei sein. Diese sind typischerweise nach dem Muster *_source.changes benannt und referenzieren das Quellpaket, aber nicht binäre .deb oder .udeb Pakete. Alle entsprechenden architekturabhängigen und architekturunabhängigen Binärpakete für alle Architekturen werden von den Build-Daemons automatisch in einer kontrollierten und vorhersehbaren Umgebung erstellt (Siehe *wanna-build* für mehr Details).

Stattdessen können Sie für viele source-only Uploads tag2upload verwenden. Das bedeutet, dass Sie nur ein signiertes Git Tag zu Salsa hochladen müssen, statt selber `.dsc`- und `.changes`-Dateien anzulegen und zu signieren. Siehe https://wiki.debian.org/tag2upload.

Es gibt einige Situationen in denen kein source-only Upload möglich ist.

Das erste Hochladen eines neuen Source Paketes (see *Neue Pakete*) muss die Binärpakte enthalten. Diese werden im Review Prozess der Debian-Archivbetreuer benötigt bevor diese zu Debian hinzugefügt werden können.

Wenn neue Binärpakte zu vorhandenen Source Paketen hinzugefügt werden dann muss das erste Hochladen was die neuen Binärpakte in `debian/control` enthält ebenfalls die Binärpkate enthalten. Der Grund hierfür ist wieder der Gleiche wie beim Einführen eines neuen Source Paketes, der Review Prozess der Debian-Archivbetreuer erfordert dies. Wünschenswert ist das Benutzen der Suite `experimental` für derartige Uploads.

Uploads, die zur Überprüfung in anderen Queues durchgeführt werden, zum Beispiel für Pakete die zu den Suites `*-backports` hinzugefügt werden sollen, können es ebenfalls erfordern, dass Binärpakte hochgeladen werden müssen.

Die Buildd Daemons starten den Bau aller Pakete aus *main* ` oder `contrib` sofern die hierzu nötigen Paketabhängigkeiten erfüllt werden können. Pakete in `non-free` und `non-free-firmware` werden nur automatisch gebaut sofern diese Pakete als geeignet markiert worden sind für das automatische Bauen (Siehe *Unfreie Pakete als automatisch erstellbar kennzeichnen*).

Die Build Daemons installieren nur zum Bau benötigte Pakete aus dem Bereich des `main` Archivs. Dies bedeutet, wenn ein Source Paket abhängige Pakete zum Bauen aus den Archivbereichen `contrib`, `non-free` oder `non-free-firmware` benötigt, dann müssen Uploads dieses Pakets vorgefertigte Binärpakete für alle unterstützen Architekturen enthalten. Definitionsbedingt kann dies nur der Fall sein wenn das Source Paket selbst in einem der Archivbereiche `contrib`, `non-free` oder `non-free-firmware` liegt.

Beim Bootstrapping einer neuen Architektur, oder einer neuen Version eines Paketes mit zirkularen Abhängigkeiten (wie z.B. ein Self Hosting Compiler) ist es auch manchmal notwendig dass das Hochladen mit den erstellten Binärpaketen erfolgen muss.

5.6.2 Hochladen nach `ftp-master`

Um ein Paket hochzuladen, sollten Sie die Dateien (einschließlich der signierten Änderungen an der dsc-Datei) mit anonymem FTP nach `ftp.upload.debian.org` in das Verzeichnis /pub/UploadQueue/ hochladen. Damit die Dateien dort verarbeitet werden, müssen sie mit einem Schlüssel aus dem Debian-Entwickler-Schlüsselbund signiert sein (siehe https://wiki.debian.org/DebianMaintainer).

Bitte beachten Sie, dass Sie die Datei "changes" zuletzt übertragen sollten. Andernfalls könnte Ihr Upload abgelehnt werden, da die Archivverwaltungs-Software die "changes"-Datei auswertet und feststellt, dass nicht alle Dateien hochgeladen wurden.

Vielleicht finden Sie auch die Debian-Pakete *dupload* oder *dput* nützlich, um Pakete hochzuladen. Diese praktischen Programme helfen den Prozess des Hochladens von Paketen nach Debian zu automatisieren.

Um Pakete aus der Upload Queue zu entfernen sehen Sie sich bitte ftp://ftp.upload.debian.org/pub/UploadQueue/README und das Debian-Paket *dcut* an.

Sie sollten aber auch über den Status Ihres Paket in `Testing` nachdenken bevor Sie eine neue Version nach `Unstable` laden. Wenn Sie eine Version in `Unstable` haben die auf die Migration nach `Testing` wartet, ist es im Allgemeinen eine gute Idee, diese zunächst migrieren zu lassen, bevor Sie eine andere neue Version hochladen. Sie sollten auch *Das Debian-Paketverfolgungssystem* auf Warnungen überprüfen, um Uploads zu vermeiden, die laufende Übergänge stören.

5.6.3 Verzögerte Uploads

Es ist manchmal nützlich, ein Paket direkt hochzuladen, aber dieses Paket erst einige Tage später im Archiv zu haben. Beispielsweise könnten Sie beim Vorbereiten eines *Non-Maintainer Uploads (NMUs)* dem Betreuer einige Tage Zeit für Reaktionen geben wollen, und dabei gleichzeitig sichergehen, dass Ihr Paket bei ausbleibender Reaktion des Betreuers im Archiv landet.

Bei einem Upload des Pakets in das Verzögerungsverzeichnis wird es in der deferred uploads queue gehalten. Wenn die angegebene Wartezeit vorüber ist, wird das Paket zur Verarbeitung in das reguläre Eingangsverzeichnis verschoben. Dies wird erledigt durch automatisches Hochladen nach `ftp.upload.debian.org` in das Upload-Verzeichnis `DELAYED/`*X*`-day` (*X* ist eine Zahl zwischen 0 und 15). 0-day wird mehrmals täglich nach `ftp.upload.debian.org` hochgeladen.

Mit dput können Sie den Parameter `--delayed` *VERZÖGERUNG* benutzen, um das Paket in eine der Warteschlangen einzureihen.

Manchmal müssen Sie ein vorher hochgeladenes Paket aus der DELAYED Queue entfernen (z.B. weil der Paket-Betreuer Sie darum gebeten hat). dcut(1) enthält Beispiele, wie das gemacht wird.

5.6.4 Sicherheits-Uploads

Laden Sie **KEIN** Paket in die Sicherheits-Upload-Warteschlange (auf `security-master.debian.org` hoch, ohne vorher eine Erlaubnis vom Sicherheits-Team erhalten zu haben. Falls das Paket nicht exakt den Anforderungen des Teams entspricht, wird es viele Probleme und Verzögerungen in der Behandlung des unerwünschten Uploads verursachen. Um Einzelheiten zu erhalten, lesen Sie *Handhabung von sicherheitsrelevanten Fehlern*.

5.6.5 Andere Upload-Warteschlangen

Es gibt in Europa eine alternative Upload-Warteschlange unter ftp://ftp.eu.upload.debian.org/pub/UploadQueue/. Sie arbeitet auf die gleiche Weise wie `ftp.upload.debian.org`, sollte aber für europäische Entwickler schneller sein.

Pakete können auch per SSH nach `ssh.upload.debian.org` hochgeladen werden. Dateien sollten in `/srv/upload.debian.org/UploadQueue` abgelegt werden. Diese Warteschlange unterstützt keine *Verzögerte Uploads*.

5.6.6 Benachrichtigungen

Die Debian-Archivbetreuer sind für die Behandlung der Paket-Uploads verantwortlich. Zum größten Teil werden Uploads automatisch täglich durch das Archiv-Verwaltungswerkzeug `dak process-upload` verarbeitet. Im Besonderen werden Aktualisierungen zu existierenden Paketen in der Distribution `Unstable` automatisch eingepflegt. In anderen Fällen, insbesondere bei neuen Paketen, wird das hochgeladene Paket manuell in die Distribution platziert. Wenn Uploads manuell behandelt werden, kann es einige Zeit dauern, bis die Änderung im Archiv erscheint. Bitte haben Sie Geduld.

Auf jeden Fall werden Sie eine E-Mail-Benachrichtigung erhalten, die anzeigt, dass das Paket dem Archiv hinzugefügt wurde und welche Fehler durch den Upload geschlossen werden. Bitte lesen Sie diese Benachrichtigung sorgfältig und prüfen Sie, ob irgendwelche Fehler, die Sie schließen wollten, nicht berücksichtigt wurden.

Die Installationsbenachrichtigung enthält außerdem die Information, in welchen Abschnitt das Paket eingefügt wird. Falls es dort eine Ungleichheit gibt, werden Sie eine separate E-Mail-Benachrichtigung darüber erhalten. Lesen Sie das Folgende.

Beachten Sie, dass, falls Sie mittels Warteschlangen hochladen, die Warteschlangen-Daemon-Software Ihnen auch per E-Mail Benachrichtigungen sendet.

Beachten Sie außerdem, dass neue Uploads im *IRC-Kanäle*-Kanal `#debian-devel-changes` publiziert werden. Falls der Upload ohne Benachrichtigung fehlgeschlagen ist, kann es sein, dass Ihr Paket keine gültige Signatur hatte. In diesem Fall finden Sie weitere Erläuterungen in `ssh.upload.debian.org:/srv/upload.debian.org/queued/run/log`.

5.7 Angabe des Paketbereichs, des Unterbereichs und der Priorität

Die Felder `Section` und `Priority` der Datei `debian/control` geben weder an, wo die Datei im Archiv tatsächlich platziert wird noch deren Priorität. Um die gesamte Integrität des Archivs zu wahren, haben die Archivbetreuer die Kontrolle über diese Felder. Die Werte in der Datei `debian/control` sind letztlich nur Hinweise.

Die Archivbetreuer behalten den Überblick über die vorschriftsmäßigen Bereiche und Prioritäten für Pakete im `override file`. Falls es dort einen Unterschied zwischen dem `override file` und den Paketfeldern, die in

`debian/control` angezeigt werden, gibt, werden Sie eine E-Mail-Benachrichtigung über die Abweichung erhalten, wenn das Paket in das Archiv installiert wird. Sie können entweder Ihre `debian/control`-Datei für Ihren nächsten Upload ändern oder eine Änderung am `override file` vorschlagen.

Um den tatsächlichen Bereich abzuändern, in den Ihr Paket abgelegt wird, müssen Sie zuerst sicherstellen, dass die Datei `debian/control` in Ihrem Paket fehlerfrei ist. Als nächstes versenden Sie einen Fehlerbericht gegen `ftp.debian.org` mit der Bitte, den Bereich oder die Priorität für Ihr Paket von dem alten auf den neuen Bereich oder die neue Priorität zu ändern. Benutzen Sie einen Betreff wie `override: PACKAGE1:section/priority, [...], PACKAGEX:section/priority` und fügen Sie die Begründung der Änderung in den Nachrichtentext des Fehlerberichts ein.

Weitere Informationen über `override files` finden Sie unter dpkg-scanpackages 1 und https://www.debian.org/Bugs/Developer#maintincorrect.

Beachten Sie, dass das Feld `Section` sowohl den Bereich als auch den Unterbereich beschreibt, die in *Bereiche* erläutert werden. Falls der Bereich "main" ist, sollte er weggelassen werden. Die Liste der erlaubten Unterbereiche finden Sie unter https://www.debian.org/doc/debian-policy/ch-archive.html#s-subsections.

5.8 Neue Upstream Versionen

Um ein Paket für eine neues Upstream-Release zu aktualisieren:

1. Lesen Sie vom Upstream-Projekt das Changelog, die NEWS und alle weitere Dokumentation, die mit der neuen Version veröffentlicht worden sein kann.
2. If possible, inspect the full diff between the old and new upstream sources, potentially filtering out uninteresting parts using filterdiff (e.g. `filterdiff -x '*.po'`). If it's too big to review thoroughly, you can use `diffstat` to get a feel for the scope and nature of the changes (and thus where new bugs may appear), and to keep an eye for anything suspicious (e.g. unexpected use of network or the appearance of dubious binary blobs).
3. Portieren Sie die alte Debian Paketierung auf die neue Version. Das beinhaltet grundsätzlich das Inkrementieren des `debian/changelog` und das Zusammenführen der `debian/patches` des alten Paketes mit dem neuen.
4. Prüfen Sie, ob Fehler behoben sind, die aktuell noch offen im BTS enthalten sind. Wenn es welche gibt, schließen Sie diese im `debian/changelog`.
5. Wenn der Patch/Merge nicht sauber ausgeführt werden kann, finden Sie heraus warum. Ein Patch kann nicht sauber ausgeführt werden, wenn er schon im neuen Upstream-Release enthalten ist oder die Upstream-Dateien, die der Patch ändert, sich substantiell geändert haben (oder gelöscht wurden).
6. Wenn es Änderungen am Build-System gab (wird mit Schritt 1 und 2 deutlich), aktualisieren Sie die `debian/rules` und `debian/control` Build-Abhängigkeiten wenn nötig.
7. Bauen Sie das neue Paket in einer isolierten Umgebung, z.B. unter Verwendung von `sbuild` oder `pbuilder`. Das sorgt dafür, dass alle nötigen Build-Abhängigkeiten in der Datei `debian/control` aufgelistet sind und eliminiert die Möglichkeit von Interferenzen mit Drittanbieter-Software auf Ihrem System.
8. Verifizieren Sie, dass das neue Pakete korrekt baut und führen Sie in dem Fall die Überprüfungen in *Das Paket testen* durch.

5.9 Fehlerbehandlung

Jeder Entwickler muss in der Lage sein, mit der Debian-Fehlerdatenbank zu arbeiten. Dies umfasst das Wissen, wie Fehlerberichte richtig eingeordnet werden (siehe *Fehler berichten*), wie sie aktualisiert und neu geordnet werden und wie sie verarbeitet und geschlossen werden.

Die Funktionen des Fehlerverfolgungssystems sind in unter Fehlerverwaltungssystem für Paket-Betreuer beschrieben. Dies umfasst das Schließen von Fehlern, Followup-Nachrichten, Zuweisen von Schweregraden, Markieren von Fehlern als weitergeleitet und andere Themen.

Operationen wie das erneute Zuweisen von Fehlern an andere Pakete, das Zusammenführen separater Fehlerberichte zum gleichen Thema oder das Wiedereröffnen von Fehlern, wenn diese voreilig geschlossen wurden, werden vom sogenannten Steuermailserver verarbeitet. Alle Befehle, die auf diesem Server verfügbar sind, werden in der Einführung in den E-Mail-Server für die Kontrolle und Manipulation beschrieben.

5.9.1 Fehlerüberwachung

Falls Sie ein guter Paketbetreuer sein möchten, sollten Sie regelmäßig die Debian-Fehlerdatenbank (BTS) für Ihre Pakete überprüfen. Das BTS enthält alle offenen Fehler Ihrer Pakete. Sie können sie prüfen, indem Sie diese Seite durchstöbern: `https://bugs.debian.org/`*nickname*`@debian.org`.

Paketbetreuer interagieren mit dem BTS über E-Mail-Adressen auf `bugs.debian.org`. Dokumentationen über verfügbare Befehle können Sie unter https://www.debian.org/Bugs/ finden oder, falls Sie das Paket `doc-debian` installiert haben, schauen Sie in die lokalen Dateien `/usr/share/doc/debian/bug-*`.

Einige finden es nützlich, regelmäßig Berichte über offene Fehler zu erhalten. Sie können einen Cron-Job wie den folgenden hinzufügen, falls Sie wöchentlich eine E-Mail erhalten möchten, die alle Fehler Ihrer Pakete zusammenfasst:

```
# ask for weekly reports of bugs in my packages
0 17 * * fri   echo "index maint address" | mail request@bugs.debian.org
```

Ersetzen Sie *address* durch Ihre offizielle Debian-Betreueradresse.

5.9.2 Auf Fehler antworten

Stellen Sie bei der Reaktion auf Fehler sicher, dass alle Diskussionen über Fehler an den ursprünglichen Übermittler des Fehlers aber auch an den Fehlerreport selbst und (wenn Sie nicht der Betreuer des Pakets sind) den Betreuer gesendet werden. Wenn Sie eine E-Mail an *123*`@bugs.debian.org` senden, wird die E-Mail an den Betreuer des Pakets gesendet und Ihre E-Mail mit im Fehlerprotokoll aufgezeichnet. Wenn Sie sich nicht an die E-Mail-Adresse des Absenders erinnern, können Sie mit *123*`-submitter@bugs.debian.org` auch den Absender des Fehlers kontaktieren. Letztere Adresse zeichnet auch die E-Mail-Korrespondenz mit im Fehlerprotokoll auf. Wenn Sie also der Betreuer des betreffenden Pakets sind, reicht es aus, die Antwort an *123*`-submitter@bugs.debian.org` zu senden. Andernfalls sollten Sie *123*`@bugs.debian.org` benutzen, damit Sie auch den Paketbetreuer erreichen.

Falls Sie einen Fehlerbericht erhalten, der FTBFS erwähnt, so bedeutet dies "Fails to build from source" (Kann nicht aus dem Quellcode erstellt werden). Portierer benutzen diese Abkürzung öfter.

Sobald Sie einen Fehlerbericht erledigt (z.B. den Fehler behoben) haben, markieren Sie ihn als `done` (dies schließt ihn), indem Sie eine Erklärung an *123*`-done@bugs.debian.org` senden. Falls Sie einen Fehler durch Ändern und Hochladen des Pakets schließen, können Sie das Schließen von Fehlern, wie in *Wann Fehler durch neue Uploads geschlossen werden* beschrieben, automatisieren.

Sie sollten Fehler *niemals* durch Senden des Befehls `close` an `control@bugs.debian.org` schließen. Falls Sie dies tun, wird der ursprüngliche Absender keine Informationen darüber erhalten, warum der Fehler geschlossen wurde.

5.9.3 Verwaltung von Fehlerberichten

Als Paketbetreuer werden Sie öfter Fehler in anderen Paketen finden oder Fehlerberichte gegen Ihre Pakete erhalten, die tatsächlich Fehler in anderen Paketen sind. Die Funktionen der Fehlerdatenbank werden in den Informationen über das Fehlerverwaltungssystem für Paket-Betreuer beschrieben. Operationen wie erneutes Zuweisen, Zusammenführen und Markieren von Fehlerberichten werden in der Einführung in den E-Mail-Server für die Kontrolle und Manipulation beschrieben. Dieser Abschnitt enthält einige Regeln für die Verwaltung Ihrer eigenen Fehler, die auf der gesammelten Erfahrung der Debian-Entwickler basieren.

Fehler für Probleme einzureichen, die Sie in anderen Paketen finden, ist eine der bürgerlichen Pflichten des Betreuerdaseins. Einzelheiten finden Sie unter *Fehler berichten*. Es ist jedoch wichtiger, die Fehler in den eigenen Paketen zu behandeln.

Hier ist eine kurze Liste der Schritte, denen Sie zu Handhabung eines Fehlerberichts folgen können:

1. Entscheiden Sie, ob der Bericht einem echten Fehler entspricht oder nicht. Manchmal rufen Anwender ein Programm nur auf die falsche Art auf, da Sie die Dokumentation nicht gelesen haben. Falls Sie dies diagnostizieren, schließen Sie den Fehler einfach und stellen Sie Informationen bereit, damit der Anwender sein Problem lösen kann (verweisen Sie auf die gute Dokumentation und so weiter). Falls der gleiche Bericht immer wieder kommt, sollten Sie sich fragen, ob die Dokumentation ausreicht oder ob das Programm den falschen Gebrauch feststellen kann, um eine aussagekräftige Fehlermeldung auszugeben. Dies ist ein Thema, das Sie mit dem Originalautor angehen sollten.

   Wenn der Fehlerübermittler mit Ihrer Entscheidung, den Fehler zu schließen nicht einverstanden ist, wird er möglicherweise erneut geöffnet, bis Sie eine Vereinbarung zur Behandlung des Fehlers gefunden haben. Wenn Sie keine Übereinkunft finden, möchten Sie möglicherweise den Fehler "wontfix" markieren, um die Leute wissen zu lassen, dass der Fehler vorhanden ist, aber nicht behoben wird. Stellen Sie sicher, dass der Fehlerübermittler die Gründe für Ihre Entscheidung versteht, indem Sie der Nachricht, in der das Tag `wontfix` hinzugefügt wird, eine Erklärung hinzufügen.

   Wenn diese Situation nicht akzeptabel ist, möchten Sie (oder der Einreicher) möglicherweise eine Entscheidung des technischen Komitees verlangen, indem Sie einen neuen Fehler gegen das Pseudopaket "tech-ctte" mit einer Zusammenfassung der Situation einreichen. Bevor Sie dies tun, lesen Sie bitte das hierzu empfohlene Verfahren.

2. Falls der Fehler zwar echt ist, jedoch ein anderes Paket betrifft, weisen Sie ihn einfach dem richtigen Paket zu. Falls Sie nicht wissen, an welches Paket er zugewiesen werden soll, sollten Sie im *IRC-Kanäle* oder auf `debian-devel@lists.debian.org` nach Hilfe fragen. Bitte informieren Sie den/die Paketbetreuer des Pakets, dem Sie den Fehler zuweisen, zum Beispiel indem Sie eine Kopie der E-Mail senden, die das erneute Zuweisen an *Paketname*`@packages.debian.org` vornimmt und erläutern Sie Ihre Beweggründe. Bitte beachten Sie, dass beim einfachen Zuweisen des Fehlers an ein anderes Paket der Betreuer dieses Pakets *keine* Nachricht davon erhält, so dass er nichts davon erfährt, bis er in die Fehlerübersicht seiner Pakete schaut.

   Falls der Fehler die Arbeit Ihres Pakets beeinflusst, denken Sie bitte daran, den Fehler zu klonen und den Klon dem Paket zuzuweisen, das das Verhalten tatsächlich verursacht. Andernfalls wird der Fehler nicht in Ihrer Liste der Paketfehler aufgeführt, was Anwender dazu veranlasst, den gleichen Fehler immer wieder zu melden. Sie sollten "Ihren" Fehler mit dem neu zugewiesenen, geklonten Fehler blockieren, um die Beziehung zu dokumentieren.

3. Manchmal müssen Sie außerdem den Schweregrad des Fehlers anpassen, so dass er der Debian-Definition entspricht. Dies geschieht deshalb, weil Leute die Dringlichkeit der Fehler erhöhen, um sicherzustellen, dass ihre Fehler rasch behoben werden. Einige Fehler können sogar auf den Schweregrad "wishlist" abgesenkt werden, wenn die angefragte Änderung nur kosmetischer Natur ist.

4. Falls der Fehler echt ist, das gleiche Problem aber bereits von jemand anderem gemeldet wurde, dann sollten die beiden relevanten Fehler mit dem Befehl "merge" des BTS zu einem zusammengeführt werden. Auf diese Art werden alle Absender des Fehlers informiert, wenn er behoben wurde. (Beachten Sie jedoch, dass E-Mails, die an den Absender eines Fehlerberichts gesandt werden, nicht automatisch an die Absender der anderen Berichte geschickt werden.) Weitere Details über die Form des "merge"-Befehls und des verwandten Befehls "unmerge" finden Sie in der Dokumentation des BTS-Steuerungs-Servers.

5. Der Absender des Fehlerberichts könnte vergessen haben, einige Informationen bereitzustellen. In diesem Fall müssen Sie die benötigten Informationen bei ihm erfragen. Sie könnten die Kennzeichnung `moreinfo` benutzen, um den Fehler entsprechend zu markieren. Außerdem können Sie den Fehler, falls sie ihn nicht reproduzieren können, als `unreproducible` kennzeichnen. Jeder, der den Fehler reproduzieren kann, ist dann eingeladen, weitere Informationen bereitzustellen, wie er reproduziert werden kann. Nach ein paar Monaten kann der Fehler geschlossen werden, falls diese Information von niemandem gesandt wird.

6. Falls sich der Fehler auf die Paketierung bezieht, beheben Sie ihn einfach. Falls Sie ihn nicht selbst beheben können, kennzeichnen Sie den Fehler mit `help`. Sie können außerdem auf `debian-devel@lists.debian.org` oder `debian-qa@lists.debian.org` um Hilfe ersuchen. Wenn es ein Upstream-Problem ist, stimmen Sie sich mit den Originalautoren ab (siehe *Abstimmung mit Originalautoren*). Falls Sie über die erforderlichen Fähigkeiten verfügen, können Sie einen Patch vorbereiten, der den Fehler behebt und ihn dem Autor mitschicken. Stellen Sie sicher, dass Sie den Patch an das BTS senden und den Fehler mit `patch` kennzeichnen.

7. Falls Sie einen Fehler in Ihrer lokalen Kopie behoben haben oder eine Fehlerbehebung in das VCS-Repository übertragen wird, können Sie den Fehler als `pending` kennzeichnen, um die Leute wissen zu lassen, dass der Fehler behoben ist und mit dem nächsten Upload geschlossen wird (fügen Sie `closes:` dem `changelog` hinzu). Dies ist besonders nützlich, falls Sie zusammen mit mehreren Entwicklern am Paket arbeiten.

8. Sobald ein korrigiertes Paket im Archiv verfügbar ist, sollte der Fehler geschlossen und die Version, in der er behoben wurde, angegeben werden. Dies kann automatisch geschehen – lesen Sie *Wann Fehler durch neue Uploads geschlossen werden*.

5.9.4 Wann Fehler durch neue Uploads geschlossen werden

Sobald Fehler und Probleme in Ihren Paketen behoben werden, liegt es in Ihrer Verantwortung als Paketbetreuer, diese Fehler zu schließen. Sie sollten jedoch keinen Fehler schließen, bis das Paket, das den Fehler schließt, im Debian-Archiv akzeptiert wurde. Daher können und sollen Sie, sobald Sie die Benachrichtigung erhalten, dass Ihr aktualisiertes Paket in das Archiv installiert wurde, den Fehler im BTS schließen. Außerdem sollte der Fehler mit der korrekten Version geschlossen werden.

Es ist jedoch möglich, das manuelle Schließen von Fehlern nach dem Upload zu vermeiden – führen Sie die behobenen Fehler in Ihrer `debian/changelog`-Datei auf. Folgen Sie dabei einer bestimmten Syntax, dann wird die Verwaltungssoftware die Fehler für Sie schließen. Zum Beispiel:

```
acme-cannon (3.1415) unstable; urgency=low

  * Frobbed with options (closes: Bug#98339)
  * Added safety to prevent operator dismemberment, closes: bug#98765,
    bug#98713, #98714.
  * Added man page. Closes: #98725.
```

Technisch gesehen beschreibt der folgende reguläre Perl-Ausdruck, wie das Schließen von Fehlern in Änderungsprotokollen identifiziert wird:

```
/closes:\s*(?:bug)?\#?\s?\d+(?:,\s*(?:bug)?\#?\s?\d+)*/ig
```

Die Syntax `closes: #`*XXX* wird bevorzugt, da sie die kürzeste Art des Eintrags ist und am einfachsten in dem Text von `changelog` integriert werden kann. Falls nicht durch den Schalter `-v` von `dpkg-buildpackage` etwas anderes angegeben wurde, werden nur die Fehler im aktuellsten Eintrag des Änderungsprotokolls geschlossen (grundsätzlich werden exakt die Fehler geschlossen, die in der Datei `.changes` im "changelog"-Teil genannt werden).

Früher wurden Uploads, die als *Non-Maintainer Uploads (NMUs)* erkannt wurden, als `fixed` statt als "closed" gekennzeichnet, aber diese Praxis wurde mit dem Beginn der Versionsverfolgung eingestellt. Das gleiche geschah mit der Markierung `fixed-in-experimental`.

Sollte es passiert sein, dass Sie eine falsche Bugnummer benutzt oder einen Bugreport in der Changelog Datei vergessen haben dann zögern Sie nicht diesen Fehler rückgängig zu machen. Um falsch geschlossene Bugreports wieder zu öffnen, senden Sie das Kommando `reopen` *XXX* an die BTS-Kontrolladresse `control@bugs.debian.org`. Um noch ausstehende Bugreports, die durch Ihren Upload gefixt worden sind, zu schließen, senden Sie die `.changes` Datei an *XXX*`-done@bugs.debian.org`, wobei *XXX* der Bugnummer entspricht. Fügen Sie Version: *YYY* in die erste Zeile gefolgt von einer Leerzeile (also insgesamt zwei Zeilen) in den Body der E-Mail, *YYY* entspricht der ersten Paketversion in der der gemeldete Fehler gefixt worden ist.

Behalten Sie im Gedächnis, dass es nicht verpflichend ist, Fehler unter Benutzung des Änderungsprotokolls zu schließen, wie oben beschrieben. Falls Sie nur einfach einen Fehler schließen möchten, der mit dem von Ihnen getätigten Upload nichts zu tun hat, können Sie dies durch Mailen einer Erläuterung an *XXX*`-done@bugs.debian.org` erledigen. Schließen Sie **keine** Fehler im Änderungsprotokolleintrag einer Version, wenn die Änderungen in dieser Version des Pakets keine Bedeutung für den Fehler haben.

Allgemeine Informationen über das Verfassen von Änderungsprotokolleinträgen finden Sie unter *Optimale Vorgehensweisen für debian/changelog*.

5.9.5 Handhabung von sicherheitsrelevanten Fehlern

Aufgrund ihrer sensiblen Natur müssen sicherheitsrelevante Fehler vorsichtig gehandhabt werden. Das Debian-Sicherheits-Team existiert, um diese Aktivitäten zu koordinieren, ausstehende Sicherheitsprobleme zu verfolgen, Paketbetreuern bei Sicherheitsproblemen zu helfen oder sie selbst zu beheben, Sicherheitsankündigungen zu versenden und `security.debian.org` zu verwalten.

Wenn Sie einen sicherheitsrelevanten Fehler in einem Debian-Paket bemerken - auch wenn Sie nicht der Betreuer sind - sammeln Sie sachdienliche Informationen über das Problem und kontaktieren Sie umgehend das Sicherheits-Team per E-Mail an `team@security.debian.org`. Sie können die E-Mail, falls gewünscht, mit dem Debian-Security-Contact-Schlüssel chiffrieren. Einzelheiten finden Sie unter https://www.debian.org/security/faq#contact. LADEN SIE **KEINE** Pakete für `Stable` hoch, ohne das Team zu kontaktieren. Nützliche Informationen sind beispielsweise:

- Ob der Fehler bereits öffentlich ist oder nicht.
- Von welchen Versionen des Pakets bekannt ist, dass sie vom Fehler betroffen sind. Prüfen Sie jede Version, die es in einem unterstützten Debian-Release gibt, ebenso wie `Testing` und `Unstable`.
- Die Art der Fehlerbehebung, falls verfügbar (Patches sind besonders hilfreich).
- Jedes korrigierte Paket das Sie für sich selbst vorbereitet haben (senden Sie das resultierende Debdiff oder alternativ nur die Dateien `.diff.gz` und `.dsc` und lesen Sie zuerst *Pakete vorbereiten, um Sicherheitsthemen anzugehen*).
- Jede Unterstützung die Sie zur Hilfe beim Testen anbieten können (Exploits, Regressionstests etc.).
- Jede Information die für die Ankündigung nötig ist (siehe *Sicherheitsankündigungen*).

Als Betreuer des Pakets sind sie verantwortlich für dessen Verwaltung, sogar im Stable-Release. Sie sind in der besten Position, um Patches zu beurteilen und aktualisierte Pakete zu testen, sehen Sie daher bitte in die folgenden Abschnitte, wie Pakete vorbereitet werden, damit sie vom Sicherheits-Team bearbeitet werden können.

5.9.5.1 Debian Security Tracker

Das Sicherheits-Team verwaltet eine zentrale Datenbank, den Debian Security Tracker. Diese enthält alle öffentlich verfügbaren Informationen, die über Sicherheitsthemen verfügbar sind: welche Pakete und Versionen betroffen oder korrigiert sind und ob daher Stable, Testing und/oder Unstable angreifbar sind. Informationen, die immer noch vertraulich sind, werden nicht zur Datenbank hinzugefügt.

Sie können diese nach einem bestimmten Thema durchsuchen, aber auch nach einem Paketnamen. Schauen Sie nach Ihrem Paket, um zu sehen, welche Themen noch offen sind. Bitte stellen Sie, falls Sie können, weitere Informationen über diese Themen bereit oder helfen Sie, diese in Ihrem Paket zu behandeln. Anweisungen finden Sie auf den Webseiten des Security Trackers.

5.9.5.2 Vertraulichkeit

Anders als bei den meisten anderen Aktivitäten innerhalb von Debian werden Informationen über Sicherheitsthemen eine Zeit lang geheim gehalten. Dies erlaubt es Software-Vertreibern, ihre Offenlegung zu koordinieren, um die Belastung ihrer Anwender zu minimieren. Ob dies der Fall ist, hängt von der Natur des Problems und der zugehörigen Fehlerbehebung ab, und ob die Angelegenheit bereits an die Öffentlichkeit durchgesickert ist.

Es gibt mehrere Möglichkeiten, wie Entwickler von einem Sicherheitsproblem erfahren:

- sie bemerken es in einem öffentlichen Forum (Maillingliste, Website etc.)
- jemand verfasst einen Fehlerbericht
- jemand informiert sie per privater E-Mail

In den ersten beiden Fällen ist die Information öffentlich und es ist wichtig, so schnell wie möglich eine Fehlerbehebung zu haben. Im letzen Fall könnte die Information nicht öffentlich sein, In diesem Fall gibt es ein paar mögliche Optionen, mit dem Problem umzugehen:

- Falls die Offenlegung der Sicherheit gering ist, ist es manchmal nicht nötig, das Problem geheim zu halten und es sollte eine Fehlerbehebung erstellt und veröffentlicht werden.

- Falls das Problem ernst ist, sollte diese Information vorzugsweise mit anderen Anbietern geteilt werden, um eine Veröffentlichung zu koordinieren. Das Sicherheits-Team hält Kontakt zu verschiedenen Organisationen und Einzelpersonen, die sich darum kümmern.

Wenn die Person, die das Problem meldet, darum bittet, dass es nicht offengelegt wird, sollte dieser Anfrage in jedem Fall Rechnung getragen werden, mit der einleuchtenden Ausnahme, das Sicherheits-Team zu informieren, damit für ein Debian-Stable-Release eine Fehlerbehebung vorbereitet werden kann. Vergessen Sie nicht, diese Tatsache zu erwähnen, wenn vertrauliche Informationen zum Sicherheits-Team gesandt werden.

Bitte beachten Sie, dass Sie keine Fehlerbehebung nach `Unstable` (oder an eine andere Stelle, wie ein öffentliches VCS-Repository) senden können, wenn Geheimhaltung nötig ist. Es reicht nicht aus, die Einzelheiten der Änderung zu verschleiern, da der Code selbst öffentlich ist und von der Allgemeinheit untersucht werden kann (und soll).

Es gibt zwei Gründe, Informationen sogar dann zu veröffentlichen, wenn um Geheimhaltung gebeten wurde: Das Problem ist bereits seit einer Weile bekannt oder es wurde ein Exploit veröffentlicht.

Das Sicherheits-Team hat einen PGP-Schlüssel, um verschlüsselte Kommunikation über sensible Themen zu ermöglichen. Einzelheiten finden Sie in der Debian Sicherheits-FAQ.

5.9.5.3 Sicherheitsankündigungen

Sicherheitsankündigungen werden nur für die aktuelle, veröffentlichte Stable-Distribution ausgegeben und *nicht* für `Testing` oder `Unstable`. Wenn Sie veröffentlicht werden, werden sie an die Mailingliste `debian-security-announce@lists.debian.org` geschickt und auf der Website für Sicherheits-Informationen veröffentlicht. Sicherheitsankündigungen werden vom Sicherheits-Team verfasst und verschickt. Es ist natürlich nicht von Nachteil, wenn ein Paketbetreuer einige Informationen dazu bereitstellen kann oder einen Teil des Textes verfasst. Informationen, die in einer Ankündigung enthalten sein sollten:

- Eine Beschreibung des Problems und seines Geltungsbereichs, einschließlich:
  - Dem Typ des Problems (Rechteausweitung, Dienstverweigerung etc.)
  - Welche Privilegien können erlangt werden und durch wen (falls zutreffend)?
  - Wie kann dies ausgenutzt werden?
  - Ist es aus der Ferne oder lokal ausnutzbar?
  - Wie wurde das Problem gelöst?

  Diese Informationen ermöglichen es Anwendern, die Bedrohung ihrer Systeme zu beurteilen.
- Versionsnummern betroffener Pakete
- Versionsnummern korrigierter Pakete
- Informationen, woher man die aktualisierten Pakete bekommen kann (üblicherweise aus dem Debian-Sicherheitsarchiv)
- References to upstream advisories, CVE identifiers, and any other information useful in cross-referencing the vulnerability

5.9.5.4 Pakete vorbereiten, um Sicherheitsthemen anzugehen

Eine Möglichkeit, dem Sicherheits-Team bei seinen Aufgaben zu helfen, besteht darin, es mit korrigierten Paketen zu versorgen, die für eine Sicherheitsankündigung für das Stable-Release geeignet sind.

Wenn eine Aktualisierung am Stable-Release vorgenommen wird, muss dabei behutsam vorgegangen werden, damit eine Änderung des Systemverhaltens vermieden wird und keine neuen Fehler eingeschleppt werden. Um dies zu erreichen, ändern Sie so wenig wie möglich, wenn Sie Fehler beheben. Anwender und Administratoren verlassen sich auf das exakte Verhalten des Releases, nachdem es veröffentlicht wurde, so dass jegliche vorgenommene Änderung das System von jemandem stören könnte. Dies trifft im Besonderen auf Bibliotheken zu: Stellen Sie sicher, dass Sie nie das API (Application Program Interface) oder das ABI (Application Binary Interface) ändern, egal wie klein die Änderung auch sein mag.

Dies bedeutet, dass das Umschwenken auf eine neue Originalversion keine gute Lösung ist. Stattdessen sollten die passenden Änderungen auf die Versionen im aktuellen Debian-Stable-Release zurückportiert werden. Generell

sind Original-Paketbetreuer bereit zu helfen, wenn nötig. Falls nicht, könnte das Debian-Sicherheits-Team in der Lage sein zu helfen.

In manchen Fällen ist es unmöglich, eine Sicherheitskorrektur zurück zu portieren, zum Beispiel, wenn große Teile des Quellcodes geändert oder überschrieben werden müssten. Falls dies der Fall ist, könnte es nötig sein, zu einer neuen Originalversion zu wechseln. Dies wird jedoch nur in extremen Situationen getan und Sie müssen dies immer vorab mit dem Sicherheits-Team abstimmen.

Darauf bezieht sich eine weitere wichtige Regel: Testen Sie immer Ihre Änderungen. Falls Sie über ein Exploit verfügen, probieren Sie es aus und kontrollieren Sie, ob es wirklich beim nicht korrigierten Paket erfolgreich ist und am korrigierten Paket scheitert. Testen Sie auch andere normale Aktionen, da eine Sicherheitskorrektur manchmal scheinbar nicht betroffene Funktionen auf subtile Weise stört.

Nehmen Sie **KEINE** Änderungen in Ihr Paket auf, die sich nicht direkt auf die Behebung der Schwachstelle beziehen. Diese müssten rückgängig gemacht werden, was Zeit kostet. Falls es in Ihrem Paket andere Fehler gibt, die Sie gerne beheben würden, machen Sie auf dem üblichen Weg ein Upload nach "proposed-updates", nachdem die Sicherheitsankündigung veröffentlicht wurde. Der Sicherheits-Aktualisierungsmechanismus ist kein Mittel, um Änderungen an Ihrem Paket einzuführen, die andernfalls vom Stable-Release abgelehnt worden wären, unterlassen sie dies also.

Überprüfen und testen Sie Ihre Änderungen so ausgiebig wie möglich. Prüfen Sie die Unterschiede zur vorherigen Version mehrmals (`interdiff` aus dem Paket `patchutils` und `debdiff` aus `devscripts` sind nützliche Werkzeuge dafür, siehe *debdiff*).

Überprüfen Sie unbedingt folgende Elemente:

- **Geben Sie die richtige Distribution als Ziel** in Ihrem `debian/changelog` an: *codename*`-security` (z.B. `trixie-security`). Geben Sie nicht *Distribution*`-proposed-updates` oder `Stable` als Ziel an!
- Verfassen Sie anschauliche, aussagekräftige Änderungsprotokolleinträge. Andere werden sich darauf verlassen, um zu bestimmen, ob ein bestimmter Fehler behoben wurde. Fügen Sie für alle eingereichten **Debian-Fehler** `closes:`-Angaben hinzu. Beziehen Sie immer einen externen Bezug ein, vorzugsweise einen **CVE-Bezeichner**, so dass Querverweise darauf möglich sind. Wenn ein CVE-Bezeichner jedoch noch nicht zugewiesen wurde, warten Sie nicht darauf, sondern fahren Sie mit dem Prozess fort. Ein späterer Querverweis auf den Bezeichner ist möglich.
- Stellen Sie sicher, dass die **Versionsnummer** angemessen ist. Sie muss größer als die des aktuellen Pakets, aber kleiner als die von Paketversionen in neueren Distributionen sein. Falls es Zweifel gibt, prüfen Sie es mit `dpkg --compare-versions`. Seien Sie vorsichtig, dass Sie keine Versionsnummer wiederverwenden, die Sie für ein vorheriges Hochladen benutzt haben, oder eine, die Konflikte mit einem binNMU auslöst. Es ist Brauch, +deb*X*u1 anzuhängen (wobei *X* die Major-Release-Nummer ist), z.B. `1:2.4.3-4+deb13u1` und natürlich muss dies bei jedem nachfolgendem Hochladen um eins zu erhöht werden.
- Sofern der Originalquellcode nicht vorher nach `security.debian.org` hochgeladen wurde (durch eine vorhergehende Sicherheitsaktualisierung), erstellen Sie den Upload **aus vollständigen Originalquellen** (`dpkg-buildpackage -sa`). Falls es einen vorhergehenden Upload nach `security.debian.org` mit der gleichen Originalversion gab, könnten Sie ohne den Originalquellcode hochladen (`dpkg-buildpackage -sd`).
- Tragen Sie Sorge, dass die **exakt gleiche *.orig.tar.{gz,bz2,xz}**-Datei wie im normalen Archiv benutzt wird. Andernfalls ist es nicht möglich, die Sicherheitskorrektur später in die Hauptarchive zu verschieben.
- Erstellen Sie das Paket auf einem **sauberen System**, auf dem nur Pakete der Distribution installiert sind, für die Sie das Paket erstellen. Falls Sie selbst nicht über ein solches System verfügen, können Sie eine debian.org-Maschine verwenden (siehe *Debian-Maschinen*) oder richten Sie ein Chroot ein (siehe *pbuilder* und *debootstrap*).

5.9.5.5 Hochladen eines korrigierten Pakets

Laden Sie **KEIN** Paket in die Sicherheits-Upload-Warteschlange (auf `security-master.debian.org`) ohne vorherige Genehmigung des Sicherheits-Teams. Falls das Paket nicht exakt den Anforderungen des Teams entspricht, wird es viele Probleme und Verzögerungen im Umgang mit dem unerwünschten Upload verursachen.

Laden Sie ihre Fehlerbehebung **NICHT** nach `proposed-updates` ohne Abstimmung mit dem Sicherheits-Team hoch. Pakete von `security.debian.org` werden automatisch direkt in das Verzeichnis `proposed-updates` kopiert. Falls bereits ein Paket mit der gleichen oder einer höheren Versionsnummer im Archiv installiert ist, wird die Sicherheitsaktualisierung durch das Archivsystem abgelehnt. Stattdessen endet auf diese Weise die Distribution Stable ohne eine Sicherheitsaktualisierung für dieses Paket.

Sobald Sie das neue Paket erstellt und getestet haben und es vom Sicherheits-Team zugelassen wurde, muss es hochgeladen werden, so dass es in den Archiven installiert werden kann. Sicherheits-Uploads werden dabei nach `ftp://security-master.debian.org/pub/SecurityUploadQueue/` hochgeladen.

Sobald ein Upload in die Sicherheitheitswarteschlange akzeptiert wurde, wird das Paket automatisch für alle Architekturen erstellt und zur Überprüfung durch das Sicherheits-Team gespeichert.

Auf Uploads, die auf Zustimmung oder Prüfung warten, kann nur das Sicherheits-Team zugreifen. Dies ist nötig, da es sich um Fehlerbehebungen für Sicherheitsprobleme handeln könnte, die noch nicht offengelegt werden dürfen.

Falls ein Mitglied des Sicherheits-Teams ein Paket akzeptiert, wird es auf `security.debian.org` installiert. Ebenso wird es für das passende *Distribution*`-proposed-updates` auf `ftp-master.debian.org` vorgeschlagen.

5.10 Paket-Updates abonnieren

Als Paket-Betreuer bekommen Sie E-Mail Benachrichtigungen für verschiedene Events (Uploads, BTS-Nachrichten, etc). Sie können solche Benachrichtigungen ebenfalls für Pakete abonnieren, die Sie nicht betreuen (oder die Sie kollavorativ mitbetreuen), in dem Sie eine E-Mail an `control@tracker.debian.org` mit `subscribe` <pkgname> im Betreff oder Body senden (mehr Details unter https://qa.pages.debian.net/distro-tracker/usage/messages.html#email-messages).

5.11 Verschieben, Entfernen, Verwaisen, Adoptieren und Wiedereinführen von Paketen

Einige Operationen zum Manipulieren von Archiven sind im Debian-Upload-Prozess nicht automatisiert. Diese Prozeduren sollten manuell durch Paketbetreuer abgewickelt werden. Dieses Kapitel gibt einen Leitfaden, was in diesen Fällen zu tun ist.

5.11.1 Pakete verschieben

Manchmal ändert ein Paket seinen Bereich. Ein Paket aus dem Bereich `non-free` könnte zum Beispiel in einer neueren Version unter der GPL erscheinen. In diesem Fall sollte es nach "main" oder "contrib" verschoben werden.[1]

Falls Sie für eines Ihrer Pakete den Bereich ändern müssen, ändern Sie die control-Information des Pakets, um es in den gewünschten Bereich zu platzieren und laden Sie das Paket erneut hoch (Einzelheiten finden Sie im Debian Policy Manual). Sie müssen sicherstellen, dass Sie Ihrem Upload die `.orig.tar.{gz,bz2,xz}`-Datei beifügen (sogar, wenn Sie keine neue Originalversion hochladen), sonst wird es nicht zusammen mit dem Rest des Pakets in dem neuen Bereich erscheinen. Falls Ihr neuer Bereich gültig ist, wird es automatisch verschoben. Falls dies nicht geschieht, wenden Sie sich an die Ftpmasters, damit Sie verstehen, was geschehen ist.

Falls Sie andererseits die `subsection` eines Ihrer Pakete ändern müssen (z.B. "devel", "admin"), ist die Prozedur etwas anders. Korrigieren Sie den in der control-Datei enthaltenen Unterbereich des Pakets und laden Sie es erneut hoch. Außerdem müssen Sie die Datei "override" aktualisieren, wie es in *Angabe des Paketbereichs, des Unterbereichs und der Priorität* beschrieben wird.

5.11.2 Pakete entfernen

Falls Sie aus irgendeinem Grund ein Paket vollständig entfernen möchten (etwa, weil es eine alte Kompatibilitätsbibliothek ist, die nicht länger erforderlich ist), müssen Sie einen Fehler gegen `ftp.debian.org` einreichen, in dem Sie darum bitten, das Paket zu entfernen. Wie alle Fehler sollte auch dieser standardmäßig den Schweregrad "normal" haben. Der Fehlertitel sollte die Form `RM:` *Paket[Architekturenliste]* `--` *Grund* haben, wobei *Paket* der

[1] Einen Leitfaden, in welchen Bereich ein Paket gehört, finden Sie im Debian Policy Manual.

Name des zu entfernenden Pakets und *Grund* eine kurze Zusammenfassung sein sollte, aus welchem Grund um Entfernen gebeten wird. *[Architekturenliste]* ist optional und wird nur benötigt, wenn das Entfernen lediglich einige Architekturen betrifft, aber nicht alle. Beachten Sie, dass `reportbug` einen Titel erstellt, der diesen Regeln entspricht, wenn Sie es benutzen, um einen Fehler des Pseudo-Pakets `ftp.debian.org` zu melden.

Falls Sie ein von Ihnen betreutes Paket entfernen wollen, sollten Sie dies im Titel des Fehlerberichts durch Voranstellen von `ROM` (Request Of Maintainer) anmerken. Es gibt mehrere andere Standardabkürzungen, die als Grund für das Entfernen von Paketen benutzt werden. Eine komplette Liste finden Sie unter https://ftp-master.debian.org/removals.html. Diese Seite stellt außerdem einen praktischen Überblick über ausstehende Anfragen zum Entfernen bereit.

Beachten Sie, dass Pakete nur aus den Distributionen `Unstable`, `Experimental` und `Stable` entfernt werden können. Pakete werden nicht direkt aus `Testing` entfernt. Sie werden vielmehr automatisch entfernt, nachdem das Paket aus `Unstable` entfernt wurde und in `Testing` kein Paket davon abhängt. (Etwas aus `Testing` zu entfernen, ist durch Einreichen eines Fehlerberichts an das Pseudopaket `release.debian.org` möglich, siehe *Entfernen aus Testing*.)

Es gibt eine Ausnahme, bei der eine explizite Anfrage zum Entfernen nicht nötig ist: Falls ein (Quell- oder Binär-) Paket nicht länger aus dem Quellcode erstellt wird, wird es halbautomatisch entfernt. Bei einem Binärpaket ist dies der Fall, wenn kein Quellpaket dieses Binärpaket weiterhin erzeugt. Falls das Binärpaket nur auf einigen Architekturen nicht länger erstellt wird, ist eine Anfrage zum Entfernen weiterhin nötig. Für ein Quell-Paket bedeutet dies, dass alle Binärpakete, die sich darauf beziehen, von einem anderen Quellpaket übernommen werden müssen.

In Ihrer Bitte um Entfernung müssen Sie detaillierte Gründe angeben, die das Entfernen rechtfertigen. Dies muss so sein, um unerwünschtes Entfernen zu vermeiden und um eine Chronik aufzubewahren, weshalb das Paket entfernt wurde. Sie können zum Beispiel den Namen des Pakets bereitstellen, dass das Entfernte ersetzt.

Üblicherweise bitten Sie nur darum, ein Paket zu entfernen, das Sie selbst betreuen. Falls Sie ein anderes Paket entfernen möchten, müssen Sie die Genehmigung seines Betreuers einholen. Sollte das Paket verwaist sein und daher keinen Betreuer haben, sollten Sie die Bitte um Entfernung zuerst auf `debian-qa@lists.debian.org` diskutieren. Falls es dort eine Übereinkunft gibt, dass das Paket entfernt werden soll, sollten Sie den bestehenden `O:`-Fehlerreport gegen das `wnpp`-Paket umbenennen und entsprechend zuweisen, anstatt einen neuen Fehlerbericht mit Bitte um Entfernen einzureichen.

Further information relating to these and other package removal related topics may be found at https://wiki.debian.org/ftpmaster_Removalsand https://qa.debian.org/howto-remove.html.

Wenn Zweifel bestehen ob ein Paket verworfen werden kann, fragen Sie per E-Mail an `debian-devel@lists.debian.org` nach Meinungen. Außerdem ist das Programm `apt-cache` aus dem Paket `apt` von Interesse. Wenn es mit `apt-cache showpkg` *Paket* aufgerufen wird, zeigt es Einzelheiten über das *Paket*, einschließlich umgekehrter Abhängigkeiten. Andere nützliche Programme umfassen `apt-cache rdepends`, `apt-rdepends`, `build-rdeps` (aus dem Paket `devscripts`) und `grep-dctrl`. Das Entfernen von verwaisten Paketen wird auf `debian-qa@lists.debian.org` diskutiert.

Sobald das Paket entfernt wurde, sollten die Fehler des Pakets behandelt werden. Sie sollten entweder im Fall, dass der tatsächliche Code in ein anderes Paket übergegangen ist, entsprechend zugewiesen werden (z.B. `libfoo12` wurde entfernt, weil `libfoo13` es ersetzt) oder geschlossen werden, falls die Software einfach nicht länger Teil von Debian ist. Wenn die Fehler geschlossen werden, sollten sie in der Version `<most-recent-version-ever-in-Debian>+rm` als behoben gekennzeichnet werden, um zu verhindern, dass sie in vorherigen Debian-Releases als behoben gekennzeichnet werden.

5.11.2.1 Entfernen von Paketen aus `Incoming`

Früher war es möglich, Pakete aus `incoming` zu entfernen. Mit der Einführung des neuen Incoming-Systems ist dies jedoch nicht mehr möglich.[4] Stattdessen müssen Sie eine neue Überarbeitung Ihres Pakets mit einer höheren Versionsnummer als der des zu ersetzenden Pakets hochladen. Beide Versionen werden im Archiv installiert, aber nur die höhere Version wird tatsächlich in `Unstable` verfügbar sein, da die vorherige sofort durch die höhere ersetzt wird. Falls Sie jedoch Ihr Paket ordnungsgemäß testen, sollte es ohnehin nicht allzu oft vorkommen, dass Sie ein Paket ersetzen.

[4] Wenn sich ein Paket jedoch noch in der Upload-Queue befindet und noch nicht nach Incomming verschoben wurde, kann es entfernt werden. (Siehe *Hochladen nach ftp-master*)

5.11.3 Umbenennen oder Ersetzen von Paketen

Wenn sich die Originalautoren eines Ihrer Pakete entscheiden, ihre Software umzubenennen (oder Ihnen beim Benennen Ihres Pakets ein Fehler unterlaufen ist), sollten Sie einen zweistufigen Prozess durchlaufen, um es umzubenennen. Im ersten Schritt ändern Sie die Datei `debian/control`, um den neuen Namen wiederzugeben, der den veralteten ersetzt, dessen Funktionalität wieder bereitstellt und zu ihm in Konflikt tritt (Einzelheiten finden Sie im Debian Policy Manual). Bitte beachten Sie, dass Sie nur dann eine `Provides`-Beziehung hinzufügen sollten, wenn alle Pakete, die von dem veralteten Paketnamen abhängen, nach dem Umbenennen weiter funktionieren. Sobald Sie das Paket hochgeladen haben und das Paket in das Archiv verschoben wurde, reichen Sie einen Fehler gegen `ftp.debian.org` ein, in dem Sie um das Entfernen des veralteten Namens ersuchen (siehe *Pakete entfernen*). Vergessen Sie nicht, gleichzeitig die Fehler passend zuzuweisen.

Eine andere Situation wäre, dass Sie einen Fehler beim Konstruieren Ihres Pakets begangen haben und wünschen, es zu ersetzen. Die einzige Möglichkeit, dies zu tun, besteht im Erhöhen der Versionsnummer und dem Hochladen dieser neuen Version. Die alte Version verliert wie üblich ihre Gültigkeit. Beachten Sie, dass dies auf jeden Teil Ihres Pakets zutrifft, einschließlich dem Quellcode: Falls Sie den Originalquell-Tarball Ihres Pakets ersetzen möchten, müssen Sie ihn mit einer veränderten Version hochladen. Eine einfache Möglichkeit ist es, `foo_1.00.orig.tar.gz` durch `foo_1.00+0.orig.tar.gz` oder `foo_1.00.orig.tar.bz2` zu ersetzen. Diese Einschränkung gibt jeder Datei auf der FTP-Seite einen einzigartigen Namen, der dabei hilft, die Einheitlichkeit über ein Netzwerk von Spiegelservern sicherzustellen.

5.11.4 Verwaisen von Paketen

Falls Sie ein Paket nicht länger betreuen können, müssen Sie andere darüber informieren und dafür sorgen, dass das Paket als verwaist gekennzeichnet wird. Sie sollten den Paketbetreuer auf `Debian QA Group <packages@qa.debian.org>` setzen und einen Fehlerbericht gegen das Pseudopaket `wnpp` senden. Der Fehlerbericht sollte im Betreff `O:` *Paket -- kurze Beschreibung* enthalten um so anzuzeigen, dass das Paket nun verwaist ist. Der Schweregrad des Fehlers sollte auf `normal` gesetzt werden falls das Paket die Priorität "standard" oder höher hat, sollte der Schweregrad auf "important" gesetzt werden. Wenn Sie es für nötig halten, senden Sie eine Kopie an `debian-devel@lists.debian.org`, indem Sie die Adresse in die Kopfzeile X-Debbugs-CC: der Nachricht einfügen (nein, benutzen Sie nicht CC:, da auf diese Art der Betreff der Nachricht die Fehlernummer nicht angibt).

Falls Sie nur die Absicht haben, das Paket abzugeben, aber im Moment noch Betreuer bleiben können, dann sollten Sie stattdessen einen Fehlerbericht gegen `wnpp` mit dem Betreff `RFA:`*Paket-- kurze Beschreibung* senden. `RFA` steht für `Request For Adoption` (Bitte um Adoption).

Weitere Informationen finden Sie auf den WNPP-Web-Seiten.

5.11.5 Adoption eines Pakets

Eine Liste von Paketen, die einen neuen Betreuer suchen, ist unter Arbeit-bedürfende und voraussichtliche Pakete (WNPP) verfügbar. Falls Sie die Verwaltung von einigen Paketen übernehmen möchten, die auf WNPP aufgeführt sind, lesen Sie bitte besagte Seite, um Informationen zu erhalten und etwas über die Prozeduren zu erfahren.

Es ist nicht in Ordnung, einfach ein Paket ohne Zustimmung des derzeitigen Betreuers zu übernehmen – das wäre Paketentführung. Sie können natürlich den aktuellen Betreuer kontaktieren und ihn fragen, ob Sie das Paket übernehmen dürfen.

Wenn ein Paket vom Betreuer jedoch vernachlässigt wurde, können Sie möglicherweise die Paketbetreuung übernehmen, indem Sie dem in *Pakete bergen* beschriebenen Paket-Bergungsprozess folgen. Wenn Sie Grund zur Annahme haben, dass ein Betreuer überhaupt nicht mehr aktiv ist, lesen Sie *Sich mit inaktiven und/oder nicht erreichbaren Paketbetreuern beschäftigen*.

Beschwerden über Betreuer sollten auf der Entwickler-Mailingliste vorgebracht werden. Falls die Diskussion mit keinem positiven Fazit endet und das Thema technischer Natur ist, erwägen Sie, den Technischen Ausschuss darauf aufmerksam zu machen. Weitere Informationen finden Sie unter Debians Technischer Ausschuss.

Wenn Sie ein altes Paket übernehmen, möchten Sie wahrscheinlich als offizieller Betreuer in der Fehlerdatenbank aufgeführt werden. Dies geschieht automatisch, sobald Sie eine neue Version mit einem aktualisierten `Maintainer`-Feld hochladen, obwohl dies nach dem Hochladen ein paar Tage dauern kann. Falls Sie für eine Weile nicht planen, eine neue Version hochzuladen, können Sie *Das Debian-Paketverfolgungssystem* benutzen,

um Fehlerberichte zu erhalten. Stellen Sie jedoch sicher, dass der alte Betreuer kein Problem damit hat, dass er während dieser Zeit weiterhin die Fehlerberichte erhalten wird.

Um den Betreuer eines Pseudo-Pakets zu ändern, senden Sie eine E-Mail an `owner@lists.debian.org` und bitten sie ihre `pseudo-packages.maintainers`-Datei anzupassen. Die E-Mail sollte den Paketnamen, den Namen und die E-Mail-Adresse des alten und neuen Betreuers und den Grund für die Änderung enthalten. Vergessen Sie nicht den alten Paket-Betreuer in den CC zu setzen.

5.11.6 Wiedereinführen vom Paketen

Pakete werden oft aufgrund veröffentlichungskritischer Fehler, fehlender Paketbetreuer, zu weniger Benutzer oder allgemein schlechter Qualität entfernt. Obwohl der Prozess der Wiedereinführung dem anfänglichen Paketierungsprozess ähnlich ist, können Sie einige Tücken umgehen, indem Sie zuerst etwas historische Recherche betreiben.

An erster Stelle sollten Sie prüfen, weshalb das Paket entfernt wurde. Diese Information können Sie unter dem Punkt Remove im News-Bereich auf der PTS-Seite des Pakets finden oder durch Durchstöbern des Protokolls unter Removed packages. Der Fehlerbericht für das Entfernen wird Ihnen sagen, weshalb das Paket entfernt wurde und einen Hinweis darauf geben, woran Sie arbeiten müssen, um das Paket wieder einzuführen. Möglicherweise gibt er auch an, dass Sie am Besten mit einer anderen Software weitermachen, anstatt das Paket wieder einzuführen.

Es ist vielleicht angebracht, die früheren Paketbetreuer zu kontaktieren, um herauszufinden, ob sie an der Wiedereinführung des Pakets arbeiten, ob sie es mitbetreuen möchten oder ob sie interessiert sind, das Paket, falls nötig, zu sponsern.

Sie sollten all die Dinge tun, die erforderlich sind, um neue Pakete einführen (*Neue Pakete*).

Sie sollten auf Basis der letzten verfügbaren Paketierung arbeiten, die sich eignet. Dies kann die letzte Version aus `Unstable` sein, die immer noch im Snapshot-Archiv vorhanden ist.

Das vom letzten Paketbetreuer benutzte Versionskontrollsystem kann nützliche Änderungen enthalten, daher ist es vermutlich eine gute Idee, dort nachzusehen. Prüfen Sie, ob die Datei `control` des vorherigen Paket irgendwelche Kopfzeilen enthält, die auf das Versionskontrollsystem des Pakets verweisen und ob es noch existiert.

Entfernen von Paketen aus `Unstable` (nicht `Testing`, `Stable` oder `Oldstable`) löst das Schließen aller Fehler aus, die sich auf das Paket beziehen. Sie sollten alle geschlossenen Fehler durchsehen (einschließlich archivierter Fehler) und diejenigen aus dem Archiv nehmen und wieder öffnen, die mit einer Version geschlossen wurden, die auf `+rm` endet und die immer noch zutreffen.

Das Entfernen von Paketen aus "Unstable" löst auch das Markieren des Pakets als entfernt im *Debian Security Tracker* aus. Debian-Mitglieder sollten entfernte Pakete als nicht Fehler behoben im Security Tracker-Repository markieren. Alle anderen sollten sich an das Sicherheitsteam wenden, um wieder eingeführte Pakete zu melden.

5.12 Portieren und portiert werden

Debian unterstützt eine immer größer werdende Anzahl von Architekturen. Sogar wenn Sie kein Portierer sind und nur eine einzige Architektur nutzen, gehört es zu Ihren Pflichten als Betreuer, die Fragen der Portierbarkeit zu kennen. Daher sollten Sie, sogar wenn Sie kein Portierer sind, das meiste in diesem Kapitel lesen.

Portieren ist das Erstellen von Debian-Paketen für Architekturen, die sich von der Originalarchitektur des Binärpakets des Paketbetreuers unterscheiden. Es ist eine einzigartige und notwendige Aktivität. Tatsächlich sind Portierer diejenigen, die die meisten Debian-Pakete kompilieren. Wenn zum Beispiel ein Paketbetreuer ein (portierbares) Quellpaket mit Binärdateien für die Architektur `i386` hochlädt, wird es für jede andere Architektur erstellt, was auf 8 weitere Builds hinausläuft.

5.12.1 Seien Sie freundlich zu Portierern

Portierer haben schwere und ungewöhnliche Aufgaben, da sie mit einer großen Zahl von Paketen umgehen müssen. Idealerweise sollte jedes Quellpaket direkt aus dem Stand korrekt erstellt werden. Unglücklicherweise ist das oft nicht der Fall. Dieser Abschnitt enthält eine Prüfliste von "Patzern", die öfter von Debian-Betreuern begangen werden – übliche Probleme, die Portierer oft in Schwierigkeiten bringen und ihre Arbeit unnötig erschweren.

Die Erste und Wichtigste ist, schnell auf einen Fehler oder ein Problem zu antworten, das ein Portierer aufgedeckt hat. Behandeln Sie Portierer höflich, als wären Sie quasi Mitbetreuer Ihres Pakets (was sie gewissermaßen sind).

Bitte seien Sie tolerant bei knappen oder sogar unklaren Fehlerberichten. Tun Sie Ihr Bestes, um Jagd auf das Problem zu machen, was immer es auch ist.

Die mit Abstand meisten Probleme, die von Portierern gefunden werden, werden durch *Paketierungsfehler* in den Quellpaketen verursacht. Hier ist eine Liste der Dinge, die Sie prüfen oder wissen sollten.

1. Stellen Sie sicher, dass Ihre `Build-Depends`- und `Build-Depends-Indep`-Einstellungen in der Datei `debian/control` richtig gesetzt sind. Die beste Methode, dies zu überprüfen, ist die Benutzung des Pakets `debootstrap`, um eine `Unstable`-Chroot-Umgebung zu erstellen (siehe *debootstrap*). Innerhalb der Chroot-Umgebung installieren Sie das Paket `build-essential` und/oder `Build-Depends-Indep`. Am Ende versuchen Sie, Ihr Paket innerhalb der Chroot-Umgebung zu erstellen. Diese Schritte können mit dem Programm `pbuilder` automatisiert werden, das vom Paket mit dem gleichen Namen bereitgestellt wird (siehe *pbuilder*).

   Falls Sie kein ordnungsgemäßes Chroot einrichten können, könnte Ihnen `dpkg-depcheck` behilflich sein (siehe *dpkg-depcheck*).

   Anweisungen zur Einrichtung von Build-Abhängigkeiten finden Sie im Debian Policy Manual.

2. Setzen Sie "architecture" auf keinen anderen Wert als `all` oder `any`, außer wenn Sie das wirklich beabsichtigen. In zu vielen Fällen folgen Paketbetreuer nicht den Anweisungen im Debian Policy Manual. Wenn Sie Ihre "architecture" nur auf eine Architektur (wie `i386` oder `amd64`) setzen, ist dies normalerweise falsch.

3. Stellen Sie sicher, dass das Quellpaket korrekt ist. Führen Sie `dpkg-source -x` *Paket*`.dsc` aus, um sicherzustellen, dass Ihr Quellpaket ordnungsgemäß entpackt wird. Dann versuchen Sie, dort Ihr Paket von Grund auf mit `dpkg-buildpackage` neu zu erstellen.

4. Stellen Sie sicher, dass Sie Ihr Quellpaket nicht mit den Dateien `debian/files` oder `debian/substvars` ausliefern. Sie sollten durch das Target `clean` von `debian/rules` entfernt werden.

5. Stellen Sie sicher, dass Sie sich nicht auf lokal installierte Pakete oder gehackte Konfigurationen oder Programme verlassen. Sie sollten zum Beispiel niemals Programme in `/usr/local/bin` oder dergleichen aufrufen. Versuchen Sie, Ihr Paket auf einem anderen Rechner zu erstellen, auch wenn dieser die gleiche Architektur hat.

6. Verlassen Sie sich nicht darauf, dass das Paket, das Sie erstellen, bereits installiert ist (ein Teilaspekt des vorherigen Problems). Es gibt natürlich Ausnahmen von dieser Regel, aber seien Sie sich bewusst, dass dies auf jeden Fall manuelles Bootstrapping erfordert und nicht durch automatisierte Paket-Builder erledigt werden kann.

7. Verlassen Sie sich, wenn möglich, nicht auf eine bestimmte Version des Compilers. Falls doch, dann stellen Sie sicher, dass Ihre Build-Abhängigkeiten diese Einschränkungen widerspiegeln, obwohl Sie sich wahrscheinlich Ärger einhandeln, da verschiedene Architekturen manchmal unterschiedliche Compiler vorgeben.

8. Sorgen Sie dafür, dass Ihre `debian/rules`-Datei separate `binary-arch`- und `binary-indep`-Targets enthält, wie es das Debian Policy Manual fordert. Stellen Sie sicher, dass beide Targets unabhängig voneinander funktionieren, damit Sie ein Target aufrufen können, ohne dass Sie vorher das andere aufgerufen haben müssen. Um dies zu prüfen, führen Sie `dpkg-buildpackage -B` aus.

9. Wenn Ihr Paket auf bestimmten Architekturen nicht unterstützt wird, sollten Sie nicht das Feld `Architecture` verwenden, um das widerzuspiegeln (es ist schwierig, das Feld richtig zu pflegen). Wenn das Paket nicht aus den Quellen gebaut werden kann, können Sie es einfach so lassen und interessierte Leute können sich die Build-Protokolle anschauen. Wenn das Paket bauen würde, sollte der Trick angewendet werden `Build-Depends` auf `unsupported-architecture [the-not-supported-arch]` zu setzen. Die Buildds werden das Paket nicht bauen, da die Build-Abhängigkeiten nicht für diese Architektur erfüllt sind. Um zu verhindern, dass für 32-Bit-Architekturen gebaut wird, kann die Build-Abhängigkeit `architecture-is-64-bit` verwendet werden. Ebenso kann `architecture-is-litte-endian` verwendet werden, um zu verhindern, dass auf Big-Endian-Systemen gebaut wird.

5.12.2 Leitlinien für Uploads von Portierern

Wenn das Paket aus dem Stand für die Architektur erstellt werden kann, auf die es portiert werden soll, haben Sie Glück und Ihre Arbeit ist einfach. Dieser Abschnitt befasst sich mit diesem Fall; er beschreibt, wie Ihr Binärpaket erstellt und hochgeladen wird, so dass es ordnungsgemäß im Archiv installiert werden kann. Falls Sie das Paket patchen müssen, um es für eine andere Architektur kompilieren zu können, führen Sie in Wirklichkeit ein Quell-NMU durch, ziehen Sie daher stattdessen *Wann und wie ein NMU durchgeführt wird* zu Rate.

Für einen Upload eines Portierers werden keine Änderungen am Quellcode vorgenommen. Sie müssen keine Dateien im Quellpaket anfassen. Dies schließt `debian/changelog` ein.

Die Vorgehensweise, `dpkg-buildpackage` aufzurufen, ist wie folgt: `dpkg-buildpackage -B -m` *E-Mail des Portierers*. Natürlich setzen Sie *E-Mail des Portierers* auf Ihre E-Mail-Adresse. Dies wird zu einem rein binären Build von nur den Paketteilen führen, die architekturabhängig sind. Dabei wird in `debian/rules` das Target `binary-arch` benutzt.

If you are working on a Debian machine for your porting efforts and you need to sign your upload locally for its acceptance in the archive, you can run `debsign` on your `.changes` file to have it signed conveniently.

5.12.2.1 Neu kompilieren oder rein binärer NMU

Manchmal ist der erste Upload einer Portierung problematisch, da die Umgebung, in der das Paket erstellt wurde, nicht gut genug war (veraltete oder hinfällige Bibliotheken, falsche Compiler etc.). Dann könnte es nötig sein, dass Sie es nur neu in einer aktualisierten Umgebung kompilieren müssen. In diesem Fall müssen Sie jedoch die Versionsnummer ändern, so dass das alte, falsche Paket im Debian-Archiv ersetzt werden kann (`dak` lehnt die Installation neuer Pakete ab, falls Sie keine höheren Versionsnummern als das aktuell verfügbare haben).

Sie müssen sicherstellen, dass Ihr rein binärer NMU das Paket nicht uninstallierbar macht. Dies könnte geschehen, wenn ein Quellpaket architekturabhängige und architekturunabhängige Pakete generiert, die unter Benutzung der ersetzbaren Dpkg-Variable `$(Source-Version)` wechselseitige Abhängigkeiten erzeugen.

Ungeachtet der nötigen Modifikation des Änderungsprotokolls werden diese Uploads als rein binäre NMUs bezeichnet – es ist nicht nötig, in diesem Fall dafür zu sorgen, dass alle anderen Architekturen sich selbst als veraltet oder eines erneuten Kompilierens bedürfig betrachten.

Solche Neu-Kompilierungen benötigen eine spezielle "magische" Versionsnummerierung, so dass die Archiv-Verwaltungswerkzeuge dies erkennen; selbst wenn es eine neue Debian-Version ist, gibt es dort keine zugehörige Aktualisierung des Quellcodes. Falls Sie dabei einen Fehler machen, werden die Archivbetreuer Ihre Aktualisierung ablehnen (aus Mangel an entsprechendem Quellcode).

Die "Magie" für ein reines Neu-Kompilierungs-NMU wird durch eine Endung ausgelöst, die an die Paketversionsnummer angehängt wird und die Form b*Zahl* hat. Wenn etwa die letzte Version, die Sie kompilierten, `2.9-3` war, sollte Ihr rein binärer NMU die Versionsnummer `2.9-3+b1` tragen. Falls die letzte Version `3.4+b1` war (d.h. ein natives Paket mit einem vorhergehenden Neu-Kompilierungs-NMU), sollte Ihr rein binärer NMU die Versionsnummer `3.4+b2`.[2] haben.

Ähnlich wie bei ersten Portierungs-Uploads ist der korrekte Weg, `dpkg-buildpackage` aufzurufen, `dpkg-buildpackage -B`, um nur die architekturabhängigen Teile des Pakets zu erstellen.

5.12.2.2 Wann Sie als Portierer ein Quell-NMU durchführen sollten

Portierer, die einen Quell-NMU durchführen, folgen generell den Leitlinien, die unter *Non-Maintainer Uploads (NMUs)* aufgeführt sind, genau wie nicht-Portierer. Es wird jedoch erwartet, dass der Wartezyklus für den Quell-NMU eines Portierers kleiner ist, als der von nicht-Portierern, da Portierer mit einer großen Zahl von Paketen zurechtkommen müssen. Die Situation variiert wiederum abhängig von der Distribution, in die hochgeladen wird. Sie variiert außerdem in Abhängigkeit davon, ob die Architektur ein Kandidat für die Integration in das nächste Stable-Release ist. Die Veröffentlichungsverwalter entscheiden, welche Architekturen Kandidaten sind und kündigen dies an.

[2] In der Vergangenheit benutzten solche NMUs die dritte Stufe im Debian-Teil der Revisionsnummer, um ihren Status als reine Neu-Kompilierung anzuzeigen. Diese Syntax war jedoch bei nativen Paketen mehrdeutig und erlaubte keine ordnungsgemäße Einordnung von reinen Neu-Kompilierungs-NMUs, Quell-NMUs und Sicherheits-NMUs im gleichen Paket. Daher wurden sie verworfen und diese neue Syntax bevorzugt.

Falls Sie als Portierer einen NMU für `Unstable` durchführen, sollten die vorher genannten Regeln der Portierung mit zwei Abwandlungen befolgt werden. Erstens ist die akzeptable Wartezeit – die Zeit zwischen dem Absenden des Fehlerberichts an das BTS und der Zeit, wenn es in Ordnung ist, einen NMU durchzuführen – sieben Tage für Portierer, die an der Distribution `Unstable` arbeiten. Diese Zeitspanne kann verkürzt werden, falls das Problem kritisch ist und eine Notlage für die Portierungsanstrengung nach Ermessen der Gruppe der Portierer besteht. (Bedenken Sie, dass nichts davon in den Debian-Richtlinien steht, die Entwickler haben sich lediglich auf gewisse Regeln diesbezüglich geeinigt.) Für Uploads nach `Stable` oder `Testing` stimmen Sie sich bitte zuerst mit dem Release-Team ab.

Zweitens sollten Portierer, die ein Quell-NMU durchführen, sicherstellen, dass der Fehler, den Sie an das BTS senden, den Schweregrad `serious` oder höher aufweist. Dies garantiert, dass ein einzelnes Quellpaket benutzt werden kann, um jede unterstützte Debian-Architektur zum Veröffentlichungszeitpunkt zu kompilieren. Es ist sehr wichtig, dass es eine Version des Quell- und Binärpakets für alle Architekturen gibt, um vielen Lizenzen zu entsprechen.

Portierer sollten versuchen, Patches zu vermeiden, die einfache Bastellösungen für Fehler in der aktuellen Version der Compiler-Umgebung, des Kernels oder der Libc enthalten. Bisweilen sind solche Bastellösungen nicht hilfreich. Falls Sie an Compiler-Fehlern und dergleichen herumbasteln müssen, stellen Sie sicher, dass Sie Ihre Arbeit ordnungsgemäß in `#ifdef` einschließen. Dokumentieren Sie außerdem Ihren Murks, damit die Leute wissen, dass er entfernt werden muss, sobald die externen Probleme behoben wurden.

Portierer könnten außerdem einen inoffiziellen Ort haben, an dem sie die Ergebnisse Ihrer Arbeit während der Wartezeit ablegen. Dies hilft anderen, die an der Portierung arbeiten, sogar während der Wartezeit aus der Arbeit des Portierers Nutzen zu ziehen. Natürlich haben solche Orte keinen offiziellen Segen oder Status, daher nehme sich der Käufer in Acht.

5.12.3 Portierungs-Infrastruktur und -Automatisierung

Es gibt eine Infrastruktur und mehrere Werkzeuge, die das Portieren von Paketen automatisieren. Dieser Abschnitt enthält einen kurzen Überblick dieser Automatisierung und Portierung mit diesen Werkzeugen. Lesen Sie die Paketdokumentation oder die Referenzen, um umfassende Informationen zu erhalten.

5.12.3.1 Mailinglisten und Web-Seiten

Web-Seiten, die den Status jeder Portierung enthalten, finden Sie unter https://www.debian.org/ports/.

Jede Portierung von Debian hat eine Mailingliste. Die Liste der Portierungs-Mailinglisten sind unter https://lists.debian.org/ports.html zu finden. Diese Listen werden benutzt, um die Arbeit der Portierer zu koordinieren und um eine Verbindungsschnittstelle der Anwender der Portierung zu den Portierern herzustellen.

5.12.3.2 Werkzeuge der Portierer

Beschreibungen von vielen Werkzeugen für die Portierung finden Sie unter *Portierungswerkzeuge*.

5.12.3.3 `wanna-build`

Das System `wanna-build` wird als verteiltes Client-/Server-Build-Verteilungssystem eingesetzt. Es wird üblicherweise zusammen mit Build-Daemons benutzt, die das Programm `buildd` ausführen. `Build daemons` sind "Slave"-Rechner, die das zentrale `wanna-build`-System kontaktieren, um eine Liste der Pakete zu beziehen, die erstellt werden müssen.

`wanna-build` ist noch nicht als Paket verfügbar. Alle Debian-Portierungsbestrebungen benutzen es jedoch zur automatisierten Paketerstellung. Das Werkzeug `sbuild`, das für die tatsächlichen Paket-Builds benutzt wird, ist allerdings als Paket verfügbar. Lesen Sie dessen Beschreibung unter *sbuild*. Bitte beachten Sie, dass sich die Paketversion auf den Build-Daemons unterscheidet, aber sie sind ähnlich genug, um Probleme nachvollziehen zu können.

`wanna-build` ist generell für Portierer nützlich. Die meisten davon erzeugten Daten sind im Web unter https://buildd.debian.org/verfügbar. Diese Daten enthalten nächtlich aktualisierte Statistiken, Warteschlangeninformationen und Protokolle von Build-Versuchen.

Debian ist ziemlich stolz auf dieses System, da es so viele Verwendungsmöglichkeiten gibt. Unabhängige Entwicklergruppen können das System benutzen, um unterschiedliche Geschmacksrichtungen von Debian, die von allgemeinem Interesse sein können oder auch nicht (zum Beispiel eine Debian-Geschmacksrichtung, die mit "bounds

checking" vom `gcc` erstellt wurde) zu erstellen. Dadurch wird Debian auch in die Lage versetzt, ganze Distributionen schnell neu zu kompilieren.

Das Wanna-Build-Team, das für Buildds verantwortlich ist, ist unter `debian-wb-team@lists.debian.org` erreichbar. Um festzustellen, wen Sie kontaktieren sollten (Wanna-Build-Team, Release-Team) und wie (per E-Mail, BTS), sei auf https://lists.debian.org/debian-project/2009/03/msg00096.html verwiesen.

Wenn Sie um BinNMUs oder Give-Backs (erneute Versuche nach gescheitertem Build) ersuchen, benutzen Sie bitte das unter https://release.debian.org/wanna-build.txt beschriebene Format.

5.12.4 Wenn Ihr Paket *nicht* portierbar ist

Einige Pakete haben immer noch Probleme mit der Erstellung und/oder Ihrer Funktion auf einigen der von Debian unterstützten Architekturen und können überhaupt nicht oder nicht in einem akzeptablen Zeitraum portiert werden. Ein Beispiel ist ein Paket, das SVGA-spezifisch ist (nur auf `i386` und `amd64` verfügbar) oder andere Hardware-spezifische Funktionen benutzt, die nicht von allen Architekturen unterstützt werden.

Um zu verhindern, dass beschädigte Pakete in das Archiv hochgeladen werden und Buildd-Zeit vergeuden, müssen Sie ein paar Dinge tun:

- Stellen Sie zuerst sicher, dass das Erstellen Ihres Pakets auf Architekturen, die es nicht unterstützt *fehlschlägt*. Es gibt mehrere Möglichkeiten dies zu bewirken. Der bevorzugte Weg ist es, während des Erstellens eine kleine Test-Suite zu verwenden, die die Funktionalität prüft und fehlschlägt, wenn es nicht funktioniert. Dies ist sowieso eine gute Idee, da es (einige) schadhafte Uploads auf allen Architekturen verhindert und außerdem ermöglicht, das Paket zu erstellen, sobald die benötigte Funktionalität verfügbar ist.

  Zusätzlich sollten Sie in `debian/control` den architecture-Wert `any` in eine Liste der unterstützten Architekturen ändern, falls Sie glauben, die Liste der unterstützten Architekturen sei ziemlich gleichbleibend. Auf diese Art wird das Erstellen ebenfalls fehlschlagen und dies einem menschlichen Leser ohne tatsächliche Versuche angezeigt.

- Um zu verhindern, dass Autobuilder unnötig versuchen, Ihr Paket zu erstellen, muss es in `Packages-arch-specific` enthalten sein, einer Liste, die vom `wanna-build`-Skript benutzt wird. Die aktuelle Version ist unter https://wiki.debian.org/PackagesArchSpecific verfügbar. Bitte lesen Sie den Anfang der Datei, wer wegen eventueller Änderungen kontaktiert werden sollte.

Bitte beachten Sie, dass es nicht ausreicht, Ihr Paket nur zu `Packages-arch-specific` hinzuzufügen, ohne dafür zu sorgen, dass das Erstellen auf nicht unterstützten Architekturen fehlschlägt: Ein Portierer oder jemand anderes, der versucht, Ihr Paket zu erstellen, könnte Ihr Paket fälschlicherweise hochladen, ohne zu bemerken, dass es nicht funktioniert. Wenn in der Vergangenheit Binärpakete für nicht unterstützte Architekturen hochgeladen wurden, bitten Sie um deren Entfernung, indem Sie einen Fehlerbericht gegen `ftp.debian.org` einreichen.

5.12.5 Unfreie Pakete als automatisch erstellbar kennzeichnen

Standardmäßig werden Pakete aus dem Bereich `non-free` und `non-free-firmware` nicht durch das Autobuilder-Netzwerk gebaut (meistens, weil die Lizenz der Pakete dem entgegen stehen könnte). Damit dort ein solches Paket dennoch gebaut wird, müssen Sie die folgenden Schritte durchführen:

1. Prüfen, ob es rechtlich erlaubt und technisch möglich ist, das Paket automatisch zu bauen;
2. `XS-Autobuild: yes` zu den Kopfzeilen von `debian/control` hinzufügen;
3. Eine E-Mail an `non-free@release.debian.org` senden und erklären, warum das Paket rechtlich und technisch automatisch gebaut werden kann.

5.13 Non-Maintainer Uploads (NMUs)

Jedes Paket hat einen oder mehrere Betreuer. Normalerweise sind das Leute, die daran arbeiten und neue Versionen des Pakets hochladen. In einigen Situationen ist es nützlich, dass auch andere Entwickler neue Versionen hochladen können. Zum Beispiel, falls sie einen Fehler in einem Paket beheben möchten, das sie nicht betreuen, oder wenn der Betreuer Hilfe benötigt, um auf Probleme zu antworten. Solche Uploads werden *Non-Maintainer Uploads (NMU)* genannt.

5.13.1 Wann und wie ein NMU durchgeführt wird

Beachten Sie die folgenden Fragen, bevor Sie einen NMU durchführen:

- Haben Sie den NMU darauf abgestimmt, dass es dem Paketbetreuer hilft? Da es Meinungsverschiedenheiten darüber geben könnte, bei was der Paketbetreuer tatsächlich Hilfe benötigt und wobei nicht, existiert die DELAYED-Warteschlange. Sie gibt dem Paketbetreuer Zeit, um zu reagieren und hat den positiven Nebeneffekt, dass unabhängige Überprüfungen des NMU-Diffs möglich sind.
- Behebt Ihr NMU wirklich Fehler? ("Fehler" umfasst jede Art von Fehlern, z.B. "wishlist"-Fehler zum Paketieren einer neuen Version der Originalautoren, es sollte jedoch Rücksicht darauf genommen werden, die Auswirkungen für den Paketbetreuer möglichst gering zu halten.) NMUs sollten nicht verwendet werden um Änderungen zu machen, die wahrscheinlich nicht einvernehmlich sind.
- Die folgenden Änderungen werden beispielsweise generell als akzeptable angesehen, außer es gibt gute Gründe den Praktiken für ein bestimmtes Paket nicht zu folgen: Verwendung des letzten veröffentlichten debhelper Kompatibilitätslevels; Verwendung von dh; Verwendung von Format 3.0 (quilt); Verwendung von lintian-brush.
- Haben Sie dem Paketbetreuer genug Zeit gegeben? Wann wurde der Fehler an das BTS gemeldet? Es ist nicht unüblich, für eine oder zwei Wochen beschäftigt zu sein. Ist der Fehler so schwer, dass er jetzt sofort behoben werden muss oder kann er noch ein paar Tage warten?
- Wie überzeugt sind Sie von Ihren Änderungen? Bitte erinnern Sie sich an den hippokratischen Eid: "Verursachen Sie vor allem keinen Schaden". Es ist besser, ein Paket mit einem offenen schweren Fehler zu belassen, als einen nicht funktionierenden Patch darauf anzuwenden oder einen, das den Fehler versteckt, anstatt Ihn zu beheben. Falls Sie nicht 100% sicher sind, was Sie getan haben, könnte es eine gute Idee sein, den Rat anderer zu suchen. Vergessen Sie nicht, das viele Leute sauer sind, wenn Ihr NMU etwas kaputt macht.
- Haben Sie Ihre Absicht, einen NMU durchzuführen, zumindest im BTS klar ausgedrückt? Falls dies zu keinen Rückmeldungen führte, ist es außerdem ratsam, zu versuchen, den Paketbetreuer auf andere Arten zu kontaktieren (private E-Mail, IRC).
- Haben Sie versucht, den Betreuer zu kontaktieren, falls er normalerweise aktiv und zugänglich ist? Im Allgemeinen sollte es als wünschenswert erachtet werden, dass sich Betreuer selbst um ein Problem kümmern und dass sie die Möglichkeit haben, Ihren Patch zu überprüfen und zu korrigieren, da sie potentielle Probleme kennen sollten, die demjenigen fehlen könnten, der ein NMU durchführt. Die Zeit wird meist besser investiert, wenn dem Betreuer die Gelegenheit gegeben wird, eine Fehlerbehebung selbst hochzuladen.

Wenn Sie einen NMU durchführen, sollten Sie zuerst dafür sorgen, dass Ihre Absicht klar ist, einen NMU durchzuführen. Dann müssen Sie einen Patch mit den Unterschieden zwischen dem aktuellen Paket und dem geplanten NMU an das BTS senden. Das Skript `nmudiff` im Paket `devscripts` könnte hilfreich sein.

Während Sie den Patch vorbereiten, sollten Sie sich über die paketspezifischen Verfahren bewusst sein, die der Betreuer möglicherweise benutzt. Diese einzubeziehen verringert dessen Aufwand, Ihre Änderungen im normalen Arbeitsablauf des Pakets zu integrieren und erhöht daher die Wahrscheinlichkeit, dass dies auch geschieht. Paket spezifische Methoden sind in `debian/README.source` beschrieben.

Sofern Sie keinen wichtigen Grund haben, dies nicht zu tun, müssen Sie dem Paketbetreuer Zeit zum Reagieren geben (zum Beispiel durch Hochladen in die `DELAYED`-Warteschlange). Hier sind einige empfohlene Werte für solche Wartezeiten:

- Der Upload behebt nur veröffentlichungskritische Fehler, die älter als sieben Tage sind, ohne Betreueraktivität beim Fehler für sieben Tage und ohne Hinweis, dass eine Fehlerbehebung im Gang ist: 0 Tage
- Upload, der nur veröffentlichungskritische Fehler behebt, die älter als sieben Tage sind: zwei Tage
- Upload, der nur veröffentlichungskritische Fehler und Fehler mit Schweregrad "important" behebt: fünf Tage
- Andere NMUs: fünfzehn Tage

Diese Verzögerungen sind nur Beispiele. In manchen Fällen, wie bei Uploads, die Sicherheitsprobleme beheben, oder bei der Behebung belangloser Fehler, die einen Übergang blockieren, ist es wünschenswert, dass ein korrigiertes Paket `Unstable` eher erreicht.

Manchmal entscheiden Veröffentlichungsverwalter, NMUs mit kürzeren Verzögerungen für eine Untermenge von Fehlern zu erlauben (z.B. veröffentlichungskritische Fehler, die älter als sieben Tage sind). Außerdem führen manche Paketbetreuer sich selbst in der Liste LowThresholdNmu (niedrige Schwellwerte für NMUs) auf und akzeptieren, dass NMUs ohne Verzögerung hochgeladen werden. Aber sogar in diesen Fällen ist es immer noch ratsam, dem Betreuer ein paar Tage Zeit zum Reagieren zu geben, bevor Sie etwas hochladen, insbesondere wenn der Patch vorher nicht im BTS verfügbar war oder falls Sie wissen, dass der Paketbetreuer allgemein aktiv ist.

Nachdem Sie einen NMU hochgeladen haben, sind Sie für mögliche Probleme verantwortlich, die Sie möglicherweise eingeleitet haben. Sie müssen das Paket im Auge behalten (eine gute Möglichkeit dafür ist *Paket-Updates abonnieren*).

Dies ist keine Lizenz, rücksichtslos NMUs durchzuführen. Falls Sie einen NMU auf den Weg bringen, während klar ist, dass die Betreuer aktiv sind und einen Patch zeitnah anerkennen würden oder falls Sie die Empfehlungen dieses Dokuments ignorieren, könnte Ihr Upload zu einem Konflikt mit dem Betreuer führen. Sie sollten immer darauf vorbereitet sein, im Nachhinein für den von Ihnen durchgeführten NMU aus eigener Kraft einstehen zu können.

5.13.2 NMUs und `debian/changelog`

Genauso wie bei jedem anderen (Quell-) Upload muss bei NMUs ein Eintrag in `debian/changelog` hinzufügt werden, der mitteilt, was mit diesem Upload geändert wurde. Die erste Zeile muss explizit erwähnen, dass dieser Upload ein NMU ist, z.B:

```
* Non-maintainer upload.
```

Die Möglichkeiten der Versionsvergabe für NMUs unterscheidet sich bei nativen und nicht nativen Paketen.

Falls es sich um ein natives Paket (ohne Debian-Revision in der Versionsnummer) handelt, muss die Versionsnummer die des letzten Betreuer-Uploads plus `+nmu`*X* sein, wobei *X* ein Zähler ist, der bei `1` beginnt. Falls der letzte Upload auch ein NMU war, wird der Zähler erhöht. Wenn beispielsweise die aktuelle Version `1.5` ist, dann hätte der NMU die Version `1.5+nmu1`.

Falls es sich um kein natives Paket handelt, sollten Sie eine untergeordnete Versionsnummer zum Debian-Revisionsteil der Versionsnummer hinzufügen (der Teil nach dem letzten Bindestrich). Diese zusätzliche Zahl muss bei `1` anfangen. Wenn zum Beispiel die aktuelle Version `1.5-2` ist, dann würde ein NMU die Version `1.5-2.1` erhalten. Falls eine neue Originalversion im NMU paketiert wird, wird die Debian-Revision auf `0` gesetzt, zum Beispiel `1.6-0.1`.

In beiden Fällen sollte der Zähler erhöht werden, falls der letzte Upload auch ein NMU war. Wenn zum Beispiel die letzte Version `1.5+nmu3` war (ein natives Paket, für das bereits ein NMU durchgeführt wurde), würde der NMU die Version `1.5+nmu4` erhalten.

Es wird ein spezielles Schema der Versionsvergabe benötigt, um zu verhindern, dass die Arbeit des Betreuers unterbrochen wird, da die Benutzung einer Ganzzahl für die Debian-Revision einen potentiellen Konflikt mit einem Betreuer-Upload hervorruft, der bereits zur Zeit des NMUs vorbereitet wird oder sogar in der Ftp-Warteschlange NEW ist. Es hat außerdem den Vorteil, dass es optisch klar erkennbar ist, wenn ein Paket im Archiv nicht vom offiziellen Betreuer stammt.

Falls Sie ein Paket nach Testing oder Stable hochladen, müssen Sie manchmal den Versionsnummernbaum "verzweigen". Dies ist zum Beispiel der Fall beim Hochladen von Sicherheitsaktualisierungen. Dazu sollte eine Version der Form `+deb`*X*`u`*Y* benutzt werden, wobei *X* die Major-Release-Nummer und *Y* eine fortlaufende, bei `1` beginnende Nummer ist. Während zum Beispiel `trixie` (Debian 13) Stable ist, hätte ein Sicherheits-NMU für Stable für ein Paket mit der Version `1.5-3` die Version `1.5-3+deb13u1`, während ein Sicherheits-NMU für `forky` die Version `1.5-3+deb14u1` erhalten würde.

5.13.3 Benutzung der Warteschlange `DELAYED/`

Nachdem Sie um Erlaubnis ersucht haben, einen NMU durchzuführen, ist es ineffizient, auf eine Antwort zu warten, da es für denjenigen, der den NMU durchführt, einen Kontextwechsel zurück zu diesem Thema erfordert. Die Warteschlange `DELAYED` (siehe *Verzögerte Uploads*) ermöglicht es einem Entwickler, einen NMU und alle nötigen Aufgaben gleichzeitig durchzuführen. Anstatt zum Beispiel dem Betreuer mitzuteilen, dass Sie das aktualisierte Paket in sieben Tagen hochladen werden, sollten Sie das Paket nach `DELAYED/7` hochladen und dem Betreuer

mitteilen, dass er sieben Tage Zeit hat, um zu reagieren. Während dieser Zeit kann der Betreuer Sie bitten, den Upload etwas länger aufzuschieben oder Ihren Upload abzubrechen.

Ein Upload kann durch *dcut* abgebrochen werden. Falls Sie `foo_1.2-1.1_all.changes` zu einer `DELAYED`-Queue hochgeladen haben, dann können Sie `dcut cancel foo_1.2-1.1_all.changes` verwenden, um den Upload abzubrechen. Die `.changes`-Datei muss hierzu nicht mehr lokal vorliegen, da Sie *dcut* angewiesen haben, eine Kommandodatei hochzuladen, durch die eine entfernt liegende Datei gelöscht werden soll. Der Name der `.changes`-Datei ist derselbe, der auch beim Upload gesendet worden ist.

Die Warteschlange `DELAYED` sollte nicht benutzt werden, um zusätzlichen Druck auf den Paketbetreuer auszuüben. Es ist besonders wichtig, dass Sie erreichbar sind, um die Verzögerung des Uploads abzubrechen, bevor die Zeit abläuft, da der Betreuer den Upload nicht selbst abbrechen kann.

Falls Sie einen NMU nach `DELAYED` durchführen und der Betreuer das Paket vor Ablauf der Verzögerung aktualisiert, wird Ihr Upload abgelehnt, da bereits eine neuere Version im Archiv verfügbar ist. Idealerweise achtet der Betreuer darauf, dass er die von Ihnen vorgeschlagenen Änderungen (oder zumindest eine Lösung für die Probleme, die sie behandeln) in diesen Upload einfließen lässt.

5.13.4 NMUs aus Sicht des Paketbetreuers

Wenn jemand einen NMU Ihres Pakets durchführt, bedeutet dies, dass er Ihnen helfen möchte, es in einem guten Zustand zu halten. Dies beschert den Anwendern schneller korrigierte Pakete. Sie könnten überlegen, ob Sie denjenigen, der den NMU durchführte, fragen möchten ob er Mitbetreuer des Pakets werden will. Der Erhalt eines NMUs für ein Paket ist keine schlechte Sache, es bedeutet nur, dass das Paket interessant genug ist, dass andere Leute daran arbeiten.

Um einen NMU anzuerkennen, schließen Sie dessen Änderungen und Änderungsprotokolleinträge in Ihren nächsten Upload ein. Falls Sie den NMU nicht anerkennen, schließen Sie den Änderungsprotokolleintrag des NMUs in Ihr Änderungsprotokoll ein, die Fehler bleiben im BTS geschlossen, werden aber als Ihre Betreuerversion des Pakets betreffend aufgeführt.

Wenn Sie jemals einen NMU zurücksetzen müssen, der eine neuere Upstreamversion paketiert hat, dann sollten Sie eine "Fake" Upstreamversion wie *AKTUELL*+`really`*VORHERIGE* benutzen bis die neueste Version erneut hochgeladen werden kann. Weitere Informationen finden Sie unter https://www.debian.org/doc/debian-policy/ch-controlfields.html#epochs-should-be-used-sparingly.

Note that the easiest way to both check if your package has been NMUed, and also automatically download and commit the changes into a git-buildpackage maintained git repository is to run `gbp import-dsc --verbose --pristine-tar apt:<package>/sid`. This example command assumes you are working on the `debian/latest` branch preparing the next upload to Debian unstable, and it assumes your `apt` has the `deb-src` line active for Debian unstable.

5.13.5 Quell-NMUs gegenüber rein binären NMUs (binNMUs)

Der vollständige Name eines NMUs ist *Quell-NMU*. Es gibt auch einen anderen Typ, der *rein binärer NMU* oder *binNMU* genannt wird. Ein binNMU ist ebenfalls ein Paket-Upload durch jemand anderes als den Paketbetreuer. Er ist jedoch rein binär.

Wenn eine Bibliothek (oder andere Abhängigkeit) aktualisiert wird, könnte es notwendig sein, das Paket neu zu erstellen. Da keine Änderungen am Quellcode nötig sind, wird das gleiche Quellpaket benutzt.

BinNMUs werden üblicherweise auf Buildds durch Wanna-Build ausgelöst. `debian/changelog` wird ein Eintrag hinzugefügt, der erklärt warum der Upload nötig war und die Versionsnummer wird, wie in *Neu kompilieren oder rein binärer NMU* beschrieben, erhöht. Dieser Eintrag sollte nicht im nächsten Upload enthalten sein.

Buildds laden Pakete für ihre Architektur als rein binäre Uploads in das Archiv hoch. Genaugenomen sind dies die binNMUs. Sie werden jedoch normalerweise nicht als NMU bezeichnet und fügen `debian/changelog` keinen Eintrag hinzu.

5.13.6 NMUs gegenüber QS-Uploads

NMUs sind Uploads von Paketen durch jemand anderes als den ihnen zugewiesenen Betreuer. Es gibt noch einen anderen Upload-Typ bei dem das hochgeladene Paket nicht dem Uploader gehört: QA-Uploads. QA-Uploads sind Uploads für verwaiste Pakete.

QS-Uploads sind normalen Betreuer-Uploads sehr ähnlich: Diese können alles korrigieren, sogar kleine Probleme. Die Versionsnummerierung ist normal und es sind keine verzögerten Uploads nötig. Der Unterschied besteht darin, dass Sie nicht als `Maintainer` oder `Uploader` des Pakets aufgeführt werden. Außerdem hat der Änderungsprotokolleintrag eines QS-Uploads eine spezielle erste Zeile:

```
* QA upload.
```

Falls Sie einen NMU durchführen möchten und es so aussieht, als sei der Betreuer nicht aktiv, ist es vernünftig zu prüfen, ob das Paket verwaist ist (diese Information wird auf der Seite des Pakets im Paketverfolgungssystem "PTS" angezeigt). Beim ersten QS-Upload zu einem verwaisten Paket sollte der Betreuer auf `Debian QA Group <packages@qa.debian.org>` gesetzt werden. Bei verwaisten Paketen, für die noch kein QS-Upload durchgeführt wurde, ist immer noch der alte Betreuer gesetzt. Eine Liste dieser Pakete finden Sie unter https://qa.debian.org/orphaned.html.

Anstatt einen QS-Upload durchzuführen, können Sie auch erwägen, das Paket zu adoptieren, indem Sie sich selbst zum Betreuer machen. Sie benötigen von niemandem eine Erlaubnis, um ein verwaistes Paket zu adoptieren, Sie müssen sich nur selbst als Betreuer einsetzen und die neue Version hochladen (siehe *Adoption eines Pakets*).

5.13.7 NMUs gegenüber Team-Uploads

Manchmal korrigieren und/oder aktualisieren Sie ein Paket, weil Sie Mitglied des Paketierungs-Teams sind (das als `Maintainer` oder `Uploader` eine Mailingliste benutzt, siehe *Gemeinschaftliche Verwaltung*), aber Sie möchten sich selbst nicht zu den `Uploaders` hinzufügen, da Sie nicht planen, regulär an diesem speziellen Paket mitzuarbeiten. Falls dies mit den Leitlinien Ihres Teams in Einklang steht, können Sie einen normalen Upload durchführen, ohne direkt als `Maintainer` oder `Uploader` aufgeführt zu werden. In diesem Fall sollten Sie Ihren Änderungsprotokolleintrag mit der folgenden Zeile beginnen:

```
* Team upload.
```

5.14 Pakete bergen

"Pakete bergen" (package salvaging) ist der Prozess, mit dem man versucht, ein Paket zu retten, das zwar offiziell nicht verwaist ist, jedoch schlecht oder gar nicht gepflegt wird. Dies ist ein schwächeres und schnelleres Verfahren, als ein Paket offiziell durch die Befugnisse des MIA-Teams zu verwaisen. Das Bergen eines Pakets ist nicht als Ersatz für das MIA-Handling gedacht und unterscheidet sich insofern, als es nichts über die generelle Aktivität eines Betreuers aussagt. Stattdessen sorgt es nur für die Übertragung des Betreuerstatus eines einzelnen Pakets, wobei alle anderen Paket-, Debian-Mitgliedschafts- oder Upload-Rechte (sofern zutreffend) unberührt bleiben.

Beachten Sie, dass der Prozess nur für die aktive Übernahme der Paketbetreuung gedacht ist. Starten Sie keine Paketbergung, wenn Sie nicht beabsichtigen, das Paket für längere Zeit zu betreuen. Wenn Sie nur bestimmte Dinge reparieren wollen, aber das Paket nicht übernehmen möchten, müssen Sie den NMU-Prozess verwenden, selbst wenn das Paket für eine Bergung berechtigt wäre. Der NMU-Prozess wird in *Non-Maintainer Uploads (NMUs)* erläutert.

Vergessen Sie nicht: Es ist inakzeptabel, Pakete anderer zu entführen. Wenn Sie sich an diesen Bergungsprozess halten, wird er Ihnen helfen sicherzustellen, dass Ihr Vorhaben keine Entführung ist, sondern eine (legale) Bergung, und Sie können dann allen Vorwürfen der Entführung mit einer Bezugnahme auf diesen Prozess entgegentreten. Dank dieses Prozesses sollten neue Beitragende keine Angst mehr haben, Pakete zu übernehmen, die vernachlässigt oder völlig vergessen wurden.

Der Prozess ist in zwei Phasen unterteilt: In der ersten Phase stellen Sie fest, ob das fragliche Paket für den Bergungsprozess *berechtigt* ist. Nur wenn die Berechtigung festgestellt wurde, können Sie in die zweite Phase, das *eigentliche* Paketbergen, eintreten.

Für weitere Informationen, Begründungen und FAQs zur Paketbergung besuchen Sie bitte die Seite Pakete bergen im Debian-Wiki.

5.14.1 Wann es berechtigt ist, ein Paket zu bergen

Ein Paket darf geborgen werden, wenn es vom aktuellen Betreuer vernachlässigt wurde. Um festzustellen, ob ein Paket vom Betreuer wirklich vernachlässigt wurde, können die folgenden Indikatoren Hinweise darauf geben:

- NMUs, insbesondere wenn es mehrere aufeinanderfolgende NMUs gegeben hat.
- Gegen das Paket eingereichte Fehlerberichte haben keine Antwort vom Betreuer.
- Die Originalautoren haben mehrere neue Versionen veröffentlicht, aber trotz dass es einen Fehlerbericht gibt, der darum bittet, die Version zu paketieren, ist dies nicht geschehen.
- Das Paket hat Qualitätssicherungsprobleme.

You will have to use your judgement as to whether a given combination of factors constitutes neglect; in case the maintainer disagrees they have only to say so (see below). If you're not sure about your judgement or simply want to be on the safe side, there is a more precise (and conservative) set of conditions in the Package Salvaging wiki page. These conditions represent a current Debian consensus on salvaging criteria. In any case you should explain your reasons for thinking the package is neglected when you file an Intent to Salvage bug later.

5.14.2 Wie man Pakete birgt

Wenn und *nur* wenn eine Paketbergung als gerechtfertigt festgestellt wurde, kann jeder potenzielle Betreuer das folgende Paketbergungsverfahren starten.

1. Öffnen Sie einen Fehler mit dem Schweregrad "important" gegen das fragliche Paket und drücken Sie in diesem Ihre Absicht aus, das Paket zu übernehmen. Dazu sollte der Titel des Fehlers mit `ITS: Paketname`[3] beginnen. Sie können alternativ anbieten, nur Mitbetreuer des Pakets zu werden. Wenn Sie den Fehler melden, müssen Sie alle Maintainer, Uploader und ggf. das Paketierungsteam explizit informieren, indem Sie sie zu `X-Debbugs-CC` hinzufügen. Falls der (die) Betreuer generell inaktiv zu sein scheinen, informieren Sie bitte das MIA-Team, indem Sie zusätzlich `mia@qa.debian.org` zu `X-Debbugs-CC` hinzufügen. Neben dem ausdrücklichen Erwähnen Ihrer Absicht zur Bergung des Paketes nehmen Sie sich bitte auch die Zeit, Ihre Einschätzung zu der Zulässigkeit des Bergens im Fehlerbericht zu dokumentieren, indem Sie beispielsweise die von Ihnen angewendeten Kriterien und zugehörige Daten anführen, um es anderen leichter zu machen, die Situation nachzuvollziehen.

2. In diesem Schritt müssen Sie warten, ob Einwände gegen die Bergung erhoben werden. Der Betreuer, jeder Mitbetreuer oder ein Mitglied des zugehörigen Paketierungsteams des betroffenen Pakets kann innerhalb von `21 Tagen` mittels einer Antwort in dem Fehlerbericht öffentlich widersprechen, wodurch der Bergungsprozess beendet wird.

   The current maintainers may also agree to your intent to salvage by filing a (signed) public response to the bug. They might propose that you become a co-maintainer instead of the sole maintainer. On team maintained packages, a member of the associated team can accept your salvaging proposal by sending out a signed agreement notice to the ITS bug, alternatively inviting you to become a new co-maintainer of the package. The team may require you to keep the package under the team's umbrella, but then may ask or invite you to join the team. In any of these cases where you have received the OK to proceed, you can upload the new package immediately as the new (co-)maintainer, without the need to utilise the `DELAYED` queue as described in the next step.

3. Wenn nach den 21 Tagen keine Antwort vom Betreuer, einem der Mitbetreuer oder dem Team an den Fehlerbericht gesendet wurde, können Sie die neue Version des Pakets mit einer minimalen Verzögerung von `sieben Tagen` in die `DELAYED`-Warteschlange hochladen. Sie sollten den ITS-Fehlerbericht über einen Eintrag im Changelog schließen. Sie müssen auch einen nmudiff an den Fehler senden, und dabei sicherzustellen, dass Kopien an den Betreuer und alle Mitbetreuer (einschließlich des Teams) des Pakets gesendet werden, indem Sie sie in der Mail an den Fehlerbericht unter `CC` eintragen.

[3] ITS it eine Abkürzung für *"Intend to Salvage"*, sinngemäß für "Absichtserklärung für das Bergen"

Während der Wartezeit in DELAYED kann der Betreuer die Bergung akzeptieren, selbst hochladen oder darum bitten, den Upload abzubrechen bzw. ihn selbst abbrechen. Die letzten beiden Fälle stoppen den Bergungsprozess, jedoch muss der Betreuer dann in dem ITS Fehlereintrag seine Aktionen begründen.

5.15 Gemeinschaftliche Verwaltung

Gemeinschaftliche Verwaltung ist ein Begriff, der die gemeinsamen Verwaltungspflichten von Debian-Paketen durch mehrere Leute beschreibt. Diese Zusammenarbeit ist fast immer eine gute Idee, da sie generell in einer höheren Qualität und einer schnelleren Fehlerbehebungszeit resultiert. Es wird dringend empfohlen, dass Pakete, die die Priorität `standard` haben, oder Teil vom Basis-Paketsatz sind, Mitbetreuer haben.

Generell gibt es einen Hauptbetreuer und einen oder mehrere Mitbetreuer. Der Hauptbetreuer ist die Person, deren Name im Feld `Maintainer` der Datei `debian/control` steht. Mitbetreuer sind alle anderen Betreuer. Sie werden normalerweise in der Datei `debian/control` im Feld `Uploaders` aufgeführt.

In seiner grundlegendsten Form ist der Prozess, neue Mitbetreuer hinzuzufügen ziemlich einfach:

- Richten Sie den Co-Maintainer mit Zugriff auf die Quellen ein, aus denen Sie das Paket erstellen. Im Allgemeinen bedeutet dies, dass Sie ein netzwerkfähiges Versionskontrollsystem wie `Git` verwenden. Salsa (siehe *salsa.debian.org: Git Repositorys und kollaborative Entwicklungsplattform*) bietet unter anderem Git-Repositories und andere kollaborative Werkzeuge.
- Fügen Sie im Feld `Uploaders` im ersten Absatz der Datei `debian/control` den korrekten Namen und die Adresse des Mitbetreuers ein.

```
Uploaders: John Buzz <jbuzz@debian.org>, Adam Rex <arex@debian.org>
```

- Die Mitbetreuer sollten das entsprechende Quellpaket abonnieren (siehe *Paket-Updates abonnieren*).

Eine andere Form der gemeinschaftlichen Verwaltung stellt die Team-Verwaltung dar. Diese wird empfohlen, falls Sie mehrere Pakete mit der gleichen Entwicklergruppe verwalten. In diesem Fall müssen die Felder `Maintainer` und `Uploaders` jedes Pakets mit Vorsicht verwaltet werden. Es wird empfohlen, eines der beiden folgenden Schemen auszuwählen:

1. Setzen Sie den Hauptverantwortlichen für das Paket in das Feld `Maintainer` ein. In `Uploaders` werden die Adresse der Mailingliste und die Team-Mitglieder, die sich um das Paket kümmern, eingetragen.
2. Setzen Sie die Adresse der Mailingliste in das Feld `Maintainer` ein. Im Feld `Uploaders` werden die Team-Mitglieder eingetragen, die sich um das Paket kümmern. In diesem Fall müssen Sie sicherstellen, dass die Mailingliste Fehlerberichte ohne menschliches Eingreifen akzeptiert (wie Moderation für Nicht-Abonnenten).

Es ist auf jeden Fall eine schlechte Idee, automatisch alle Team-Mitglieder in das Feld `Uploaders` einzutragen. Es überfüllt die Paketübersicht des Entwicklers (siehe *Paketübersicht des Entwicklers*) mit Paketen, um die sich nicht wirklich jemand kümmert und erweckt den falschen Eindruck einer guten Betreuung. Aus dem gleichen Grund müssen sich Team-Mitglieder nicht selbst im Feld `Uploaders` hinzufügen, nur weil sie das Paket einmal hochgeladen haben. Sie können einen Team-Upload durchführen (siehe *NMUs gegenüber Team-Uploads*). Im umgekehrten Fall ist es eine schlechte Idee, ein Paket nur mit der Adresse der Mailingliste im Feld `Maintainer` und ohne einen Eintrag in `Uploaders` zu belassen.

5.16 Die Distribution Testing

5.16.1 Grundlagen

Pakete werden üblicherweise in der Distribution `Testing` installiert, nachdem sie in `Unstable` gewissen Tests unterzogen wurden.

Sie müssen auf allen Architekturen synchron sein und dürfen keine Abhängigkeiten haben, die sie nicht installierbar machen. Außerdem dürfen sie zum Zeitpunkt, an dem sie in `Testing` installiert werden, keine bekannten veröffentlichungskritischen Fehler haben. Auf diese Art sollte `Testing` immer ein potentieller Veröffentlichungskandidat sein. Bitte lesen Sie das Folgende, um weitere Einzelheiten zu erfahren.

5.16.2 Aktualisierungen von Unstable

Die Skripte, die die Distribution `Testing` aktualisieren, werden zweimal täglich ausgeführt, gleich nach der Installation der aktualisierten Pakete. Diese Skripte werden `britney` genannt. Sie generieren die `Packages`-Dateien für die Distribution `Testing`, aber dabei versuchen diese entsprechend clever zu agieren, sie versuchen jede Unstimmigkeit zu vermeiden und nur fehlerfreie Pakete zu benutzen.

Die Aufnahme eines Pakets von `Unstable` ist von folgendem abhängig:

- Das Paket muss in `unstable` für eine bestimmte Anzahl an Tagen verfügbar sein (Siehe *Auswahl Dringlichkeit des Uploads*). Bitte beachten Sie, dass die Dringlichkeit „sticky" ist, was bedeutet, dass die höchste Dringlichkeitsstufe, die seit dem vorherigen Übergang zu *testing* ` hochgeladen wurde, berücksichtigt wird.
- Es darf keine veröffentlichungskritischen Fehler haben (veröffentlichungskritische Fehler betreffend die in `Unstable` verfügbare Version, aber nicht die in `Testing`).
- Es muss auf allen Architekturen verfügbar sein auf denen es vorher in `Unstable` erstellt wurde. Um diese Information zu prüfen, könnte *Das Hilfswerkzeug dak ls* von Interesse sein.
- Es darf keine Abhängigkeiten von Paketen zerstören, die bereits in `Testing` verfügbar sind.
- Die Pakete, von denen es abhängt, müssen entweder in `Testing` verfügbar sein oder sie müssen zur gleichen Zeit in `Testing` akzeptiert werden (und das werden sie, falls sie alle nötigen Kriterien erfüllen).
- Die aktuelle Phase des Projekts, d.h. zum Beispiel, dass automatische Übergänge während des *Freeze* der Distribution `Testing` ausgesetzt werden.

Um herauszufinden ob ein Paket den Übergang nach `Testing` erfolgreich durchläuft oder nicht, schauen Sie in die Ausgabe des `Testing`-Skripts auf der Web-Seite Debian "Testing"-Distribution oder benutzen Sie das Programm `grep-excuses` aus dem Paket `devscripts`. Dieses Hilfswerkzeug kann einfach in einer crontab 5 benutzt werden, um sich selbst auf dem aktuellen Stand über den Fortschritt des Pakets nach `Testing` zu informieren.

Die Datei `update_excuses` gibt nicht immer den genauen Grund an, warum ein Paket abgelehnt wurde. Sie können es selbst herausfinden, indem Sie schauen, was durch die Aufnahme des Pakets zerstört werden würde. Die Web-Seite Debian "Testing"-Distribution stellt einige weitere Informationen über die üblichen Probleme bereit, die derartigen Ärger verursachen.

Manchmal errreichen einige Pakete `Testing` niemals, da die Zusammensetzung wechselseitiger Beziehungen zu kompliziert ist und durch das Skript nicht aufgelöst werden können. Im Folgenden finden Sie Einzelheiten.

Einige weitere Abhängigkeitsanalysen werden unter https://release.debian.org/migration/ dargestellt — aber seien Sie gewarnt, diese Seite zeigt auch Build-Abhängigkeiten, die von Britney nicht berücksichtigt werden.

5.16.2.1 Veraltet

Für das `Testing` Migrationsskript bedeutet veraltet: Es gibt verschiedene Versionen in``Unstable`` für die Release-Architekturen (mit Ausnahme der Architekturen in outofsync_arches, outofsync_arches ist eine Liste von Architekturen, die nicht mithalten ist (in `britney.py`), derzeit aber leer ist). Veraltet hat überhaupt nichts mit den Architekturen zu tun, die dieses Paket in `Testing` hat.

Sehen Sie sich dieses Beispiel an:

	alpha	arm
testing	1	-
unstable	1	2

Das Paket ist auf `alpha` in `Unstable` veraltet und wird nicht nach `Testing` gelangen. Das Paket zu entfernen würde überhaupt nicht helfen. Das Paket ist immer noch auf `alpha` veraltet und wird nicht nach `Testing` gelangen.

Falls FTP-Master jedoch ein Paket in `Unstable` entfernt (hier auf `arm`):

	alpha	arm	hurd-i386
testing	1	1	-
unstable	2	-	1

In diesem Fall ist das Paket auf allen Architekturen in `Unstable` aktuell (und das zusätzliche `hurd-i386` tut nichts zur Sache, da es keine Release-Architektur ist).

Manchmal kommt die Frage auf, ob es möglich ist, Pakete aufzunehmen, die noch nicht auf allen Architekturen erstellt wurden: Nein. Einfach nur nein. (Außer Sie betreuen glibc oder ähnlich).

5.16.2.2 Entfernen aus Testing

Manchmal wird ein Paket entfernt, um einem anderen Paket die Aufnahme zu gewähren. Dies geschieht nur, um einem *anderen* Paket die Aufnahme zu gewähren, falls dies sinnvoll ist. Angenommen, `a` könnte z.B. nicht mit der neuen Version von `b` installiert werden, dann könnte `a` entfernt werden, um `b` die Aufnahme zu ermöglichen.

Natürlich gibt es einen anderen Grund, ein Paket aus `Testing` zu entfernen. Es ist einfach zu fehlerhaft (und ein einfacher veröffentlichungskritischer Fehler reicht dafür aus).

Wenn ein Paket außerdem aus `Unstable` entfernt wurde und kein Paket in `Testing` mehr davon abhängt, dann wird es automatisch entfernt.

5.16.2.3 Zirkulare Abhängigkeiten

Eine Situation mit der Britney nicht gut klar kommt ist, wenn Paket `a` von einer neuen Version des Pakets `b` abhängt und umgekehrt.

Ein Beispiel hierfür:

	testing	unstable
a	1; depends: b=1	2; depends: b=2
b	1; depends: a=1	2; depends: a=2

Weder Paket `a` noch Paket `b` wird für die Aktualisierung berücksichtigt.

Aktuell erfordert dies einige manuelle Eingriffe des Release-Teams. Bitte kontaktieren Sie es per E-Mail an `debian-release@lists.debian.org`, falls dies bei einem Ihrer Pakete auftritt.

5.16.2.4 Beeinflussen eines Pakets in Testing

Im Allgemeinen gibt es nichts, was vom aktuellen Status eines Pakets in `Testing` von Bedeutung für den Übergang der nächsten Version von `Unstable` zu `Testing` ist, mit zwei Ausnahmen: Wenn die Anzahl der Menge an RC-Fehlern des Pakets verkleinert wird, kann das Paket migrieren auch wenn es noch veröffentlichungskritische Fehler hat. Die zweite Ausnahme ist die wenn die Version des Pakets in `Testing` auf den verschiedenen Architekturen nicht synchron ist. Dann kann jede Architektur auf die Version des Quellpakets aktualisiert werden. Dies kann jedoch nur geschehen, wenn das Paket zuvor zwangsmäßig migriert wurden ist, die Architektur in der Datei outof-sync_arches befindet oder während der Migration nach `Testing` überhaupt kein Binärpaket dieser Architektur in `Unstable` vorhanden war.

Zusammengefasst heißt das: Der einzige Einfluss eines Pakets, das sich in `Testing` befindet, auf eine neue Version des gleichen Pakets besteht darin, dass die neue Version leichter aufgenommen werden kann.

5.16.2.5 Einzelheiten

Falls Sie die Einzelheiten interessieren, hier ein paar Erklärungen wie Britney funktioniert:

Die Pakete werden betrachtet, um festzulegen, ob Sie gültige Kandidaten sind. Daraus werden die update excuses (Gründe für eine Nicht-Aktualisierung) erzeugt. Die häufigsten Gründe warum ein Paket nicht berücksichtigt wird lauten: zu neu, zu viele veröffentlichungskritische Fehler oder auf einigen Architekturen veraltet. Für diesen Teil

von Britney verfügen die Veröffentlichungsverwalter über Druckmittel verschiedener Stärke (Hinweise genannt, siehe unten), um eine Berücksichtigung des Pakets durch Britney zu erzwingen.

Nun kommt der komplexere Teil: Britney versucht, `Testing` mit den gültigen Kandidaten zu aktualisieren. Dazu versucht Britney jeden gültigen Kandidaten zur Distribution `Testing` hinzuzufügen. Falls die Zahl nicht installierbarer Pakete in `Testing` sich nicht erhöht, wird das Paket akzeptiert. Ab diesem Zeitpunkt wird das Paket als Teil von `Testing` betrachtet, so dass alle anschließenden Tests zur Installierbarkeit dieses Paket einbeziehen. Hinweise des Release-Teams werden, abhängig vom genauen Typ, vor diesem Hauptdurchlauf verarbeitet.

Falls Sie weitere Einzelheiten suchen, können Sie unter https://release.debian.org/britney/update_output/ nachsehen.

Die Hinweise sind unter https://release.debian.org/britney/hints/ verfügbar. Dort finden Sie auch die Beschreibung dazu. Mit den Hinweisen kann das Debian Release-Team Pakete blockieren oder entsperren, Paketmigration nach `Testing` vereinfachen oder auch erzwingen, Pakete aus `Testing` entfernen, Uploads genehmigen für *Direkte Aktualisierungen für Testing* oder die Dringlichkeit überschreiben.

5.16.3 Direkte Aktualisierungen für Testing

Die Distribution `Testing` wird, den genannten Regeln folgend, mit Paketen aus `Unstable` gespeist. In einigen Fällen ist es jedoch nötig, Pakete hochzuladen, die nur für `Testing` erstellt wurden. Dafür empfiehlt es sich, nach `testing-proposed-updates` hochzuladen.

Merken Sie sich, dass dorthin hochgeladene Pakete nicht automatisch verarbeitet werden, sie müssen erst durch die Hand des Veröffentlichungsverwalters gehen. Daher sollten sie besser über einen triftigen Grund verfügen, dorthin hochzuladen. Um zu erfahren, was in den Augen der Veröffentlichungsverwalter ein triftiger Grund ist, sollten sie die Anweisungen lesen, die diese reglemäßig auf `debian-devel-announce@lists.debian.org` verteilen.

Sie sollten nicht auf `testing-proposed-updates` hochladen, wenn Sie Ihre Pakete über `Unstable` aktualisieren können. Wenn Sie dies nicht können (z. B. weil Sie eine neuere Entwicklungsversion in `Unstable` haben), dann können Sie diese Option verwenden. Selbst wenn ein Paket eingefroren ist, sind Aktualisierungen über `Unstable` möglich, wenn beim Hochladen über `Unstable` keine neuen Abhängigkeiten entstehen.

Versionsnummern werden üblicherweise durch Anhängen von +deb*X*u*Y* ausgewählt, wobei *X* die Major-Release-Nummer von Debian und *Y* eine bei 1 beginnende Nummer ist, die hochgezählt wird, z.B. `1:2.4.3-4+deb13u1`.

Bitte stellen Sie sicher, dass Sie alle folgenden Punkte für Ihrem Upload beachtet haben:

- Vergewissern Sie sich, dass Ihr Paket wirklich `testing-proposed-updates` durchlaufen muss und nicht über `Unstable` gehen kann.
- Achten Sie darauf, dass Sie nur die kleinstmögliche Menge von Änderungen eingefügt haben.
- Sorgen Sie dafür, dass das Änderungsprotokoll eine entsprechende Erklärung enthält.
- Überzeugen Sie sich, dass Sie den *Codenamen der Veröffentlichungen* von Testing (z.B. `forky`) als Zieldistribution eingetragen haben.
- Prüfen Sie nach, ob Sie Ihr Paket in `Testing` und nicht in `Unstable` getestet haben.
- Gehen sie sicher, dass Ihre Versionsnummer höher als die Version in `Testing` und `testing-proposed-updates` ist und niedriger als in `Unstable`.
- Fragen Sie im Vorfeld die Veröffentlichungsmanager nach einer Erlaubnis.
- Nach dem Hochladen und erfolgreichen Erstellen auf allen Plattformen kontaktieren Sie das Release-Team unter `debian-release@lists.debian.org` und ersuchen Sie es um Genehmigung Ihres Uploads.

5.16.4 Häufig gestellte Fragen

5.16.4.1 Was sind veröffentlichungskritische Fehler und wie werden diese gezählt?

Alle Fehler mit einem höheren Schweregrad werden standardmäßig als veröffentlichungskritisch angesehen. Aktuell sind dies Fehler der Schweregrade `critical`, `grave` und `serious`.

Von solchen Fehlern wird angenommen, dass sie einen Einfluss darauf haben, ob das Paket mit dem Stable-Release von Debian veröffentlicht wird: Im Allgemeinen würde ein Paket, das offene veröffentlichungskritische Fehler hat, nicht nach Testing gelangen und demzufolge nicht in Stable veröffentlicht werden.

Als Unstable-Fehleranzahl werden alle veröffentlichungskritischen Fehler gezählt, die für eine *Paket-/Versions*-Kombination markiert sind, welche in Unstable für eine Release-Architektur verfügbar ist. Die Fehleranzahl in Testing ist analog dazu definiert.

5.16.4.2 Wie kann das Installieren eines Pakets in Testing andere Pakete möglicherweise beschädigen?

Die Struktur der Distributionsarchive ist so aufgebaut, dass Sie nur eine Version eines Pakets enthalten kann. Ein Paket wird durch seinen Namen definiert. Wenn also das Quellpaket acmefoo zusammen mit seinen Binärpaketen acme-foo-bin, acme-bar-bin, libacme-foo1 und libacme-foo-dev nach Testing installiert wird, wird die alte Version entfernt.

Die alte Version könnte jedoch ein Binärpaket mit einem alten Soname einer Bibliothek bereitstellen, wie libacme-foo0. Das Entfernen der alten acmefoo wird libacme-foo0 entfernen, was alle Pakete beschädigt, die davon abhängen.

Offenbar betrifft dies hauptsächlich Pakete, die sich ändernde Zusammenstellungen von Binärpaketen in unterschiedlichen Versionen bereitstellen (wiederum hauptsächlich Bibliotheken). Es wird jedoch außerdem Pakete betreffen, deren Versionsabhängigkeiten über die Varianten ==, <= oder << deklariert wurden.

Wenn sich die Zusammenstellung von Binärpaketen, die von einem Quellpaket bereitgestellt werden, auf diese Weise ändert, müssen alle Pakete, die von den alten Programmen abhängen, aktualisiert werden, damit sie stattdessen von den neuen Programmen abhängen. Da das Installieren eines solchen Quellpakets in Testing alle Pakete beschädigt, die in Testing davon abhängen, ist nun auf Folgendes zu achten: All die abhängigen Pakete müssen aktualisiert werden und selbst bereit zur Installation sein, so dass sie nicht beschädigt werden. Sobald alles bereit ist, ist normalerweise ein manuelles Eingreifen des Veröffentlichungsverwalters oder eines Assistenten nötig.

Falls Sie Probleme mit wie hier dargestellten komplizierten Gruppen von Paketen haben, kontaktieren Sie debian-devel@lists.debian.org oder debian-release@lists.debian.org, um Hilfe zu erhalten.

5.17 Das Archive von Stable Backports

5.17.1 Grundlagen

Wenn ein Paket die Distribution testing erreicht hat dann ist es für jede Person in Debian mit Upload Berechtigungen (Siehe nachstehende Infos) möglich einen Backport dieses Paketes zu bauen und nach stable-backports hoch zu laden, dies ermöglicht eine einfache Installation eines Paketes aus testing auf einem System was der stable Distribution folgt.

Man sollte eine Version eines Pakets erst dann nach stable-backports hochladen, wenn die passende Version bereits das testing Archiv erreicht hat.

5.17.2 Ausnahmen von der erst-testing Regel

Die einzige Ausnahme von der oben genannten Regel besteht, wenn es einen wichtigen Sicherheitsfix gibt, der einen schnellen Upload rechtfertigt: In einem solchen Fall besteht keine Notwendigkeit, den Upload des Sicherheitsfixes in das stable-backports Archiv zu verzögern. Es wird jedoch dringend empfohlen, das Paket zunächst in „unstable" zu reparieren, bevor ein Fix in das stable-backports Archiv hochgeladen wird.

5.17.3 Wer kann Pakete im Stable-Backports Archiv verwalten?

Es ist nicht unbedingt Aufgabe des ursprünglichen Paketbetreuers, die stable-backports Version des Pakets zu pflegen. An sich kann dies jeder tun und man braucht dazu nicht einmal die Genehmigung des ursprünglichen Paketbetreuers. Es empfiehlt sich jedoch, sich zunächst mit dem ursprünglichen Betreuer des Pakets in Verbindung zu setzen, bevor Sie versuchen, mit der Portierung eines Pakets in stable-backports zu beginnen. Der Paketbetreuer kann den Wunsch haben selbst den Backport selbst zu tätigen, oder Ihnen helfen den Backport zu

erstellen. Es ist beispielsweise nicht ungewöhnlich, einen Patch auf die instabile Version eines Pakets anzuwenden, um dessen Rückportierung zu erleichtern.

5.17.4 Wann kann man damit beginnen Pakete nach stable-backports hoch zu laden?

Ein neues `stable-backports` wird vor dem Freeze der nächsten `stable` Suite erstellt. Allerdings ist ein Hochladen dorthin bis zum Ende des Freeze-Zyklus nicht gestattet. Das `stable-backports` Archiv wird normalerweise einige Wochen vor der endgültigen Veröffentlichung der nächsten `stable` Suite geöffnet, es macht jedoch keinen Sinn etwas hoch zu laden, bis die Veröffentlichung tatsächlich erfolgt ist.

5.17.5 Wie lange muss ein Paket betreut werden wenn es nach stable-backports geladen worden ist?

Das `stable-backports` Archiv wird während des gesamten Lebenszyklus der Debian `stable` Suite zur Pflege von Fehler- und Sicherheitsprobleme betreut. Daher impliziert ein Upload auf `stable-backports` die Bereitschaft, das zurückportierte Paket für die Dauer der `stable` Suite zu betreuen, was voraussichtlich etwa drei Jahre nach der ersten Veröffentlichung sind.

Die Person, die nach Backports hochlädt, sollte sich auch für Security Fixes der zurückportierten Pakete während der Lebensdauer von „stable" verantwortlich sehen.

Es ist zu beachten, dass `stable-backports` nicht Teil der LTS oder ELTA Initiative ist. Die FTP-Master von `stable-backports` werden das Repository `stable-backports` für Uploads schließen sobald `stable` das Ende das Lebenszyklus (EOF) erreicht hat (Z.B. wenn `stable` nur noch durch das LTS-Team verwaltet wird). Dadurch erfolgt keine weitere Bearbeitung von Paketen in `stable-backports` nach dem offiziellen Ende das Lebenszykluses der Suite `stable` da keine neueren Uploads mehr akzeptiert werden.

5.17.6 Wie oft sollte man ein Paket nach stable-backport hochladen?

Die Pakete in Backports sollen den Entwicklungen im Testing folgen. Daher wird erwartet, dass jede bedeutende Aktualisierung in „testing" einen Upload in „stable-backports" auslöst, bis das neue „stable" Release veröffentlicht wird. Bitte portieren Sie kleinere Versionsänderungen jedoch nicht ohne für den Benutzer sichtbare Änderungen oder Fehlerbehebungen nach Backports.

5.17.7 Wie kann man mehr über das Zurückportieren erfahren?

Auf der Webseite Wie man beitragen kann können Sie mehr zu dieser Frage erfahren.

Es wird auch empfohlen die Frequently Asked Questions (FAQ) zu lesen.

KAPITEL 6

Optimale Vorgehensweise beim Paketieren

Debians Qualität ist größtenteils den Debian-Richtlinien zu verdanken, die explizit grundlegende Anforderungen definieren, die alle Debian-Pakete erfüllen müssen. Außerdem stehen gemeinsame Erfahrungen im Paketieren aus der Vergangenheit hinter den Debian-Richtlinien. Viele sehr talentierte Leute haben großartige Werkzeuge geschaffen, Werkzeuge, die Ihnen als Debian-Betreuer helfen, ausgezeichnete Pakete zu erstellen und zu pflegen.

Dieses Kapitel stellt einige optimale Vorgehensweisen für Debian-Entwickler vor. Das alles sind lediglich Empfehlungen und keine Anforderungen oder feste Regeln. Es sind nur subjektive Hinweise, Ratschläge und Fingerzeige, die von Debian-Entwicklern gesammelt wurden. Suchen Sie sich einfach das heraus, was Ihnen am meisten zusagt.

6.1 Optimale Vorgehensweisen für `debian/rules`

Die folgenden Empfehlungen gelten für die Datei `debian/rules`. Da `debian/rules` den Build-Prozess steuert und die Dateien auswählt, die in das Paket gelangen (direkt oder indirekt), ist es normerweise die Datei, der die Betreuer die meiste Zeit widmen.

6.1.1 Helferskripte

Der Grund für die Benutzung von Helferskripten in `debian/rules` ist, dass sie den Betreuern eine gemeinsame allgemeine Logik inmitten vieler Pakete ermöglichen. Nehmen Sie zum Beispiel die Frage, wie Menü-Einträge installiert werden: Sie müssen die Datei in `/usr/share/menu` (oder `/usr/lib/menu` für ausführbare binäre Menü-Dateien, wenn nötig) ablegen und den Betreuerskripten Befehle hinzufügen, um Menü-Einträge zu registrieren bzw. ihre Registrierung zu entfernen. Dies ist eine sehr häufige Tätigkeit für Pakete. Warum sollte daher jeder Betreuer all dies für sich selbst neu schreiben und dabei möglicherweise Fehler verursachen? Außerdem, gesetzt den Fall, das Menü-Verzeichnis würde sich ändern, müsste dann jedes Paket geändert werden.

Helferskripte kümmern sich um diese Probleme. Angenommen, Sie erfüllen alle Gepflogenheiten, die das Helferskript erwartet, dann kümmert sich das Helferskript um alle Einzelheiten. Änderungen an den Richtlinien können im Helferskript erledigt werden. Dann müssen Pakete nur mit der neuen Version des Helferskripts erstellt und sonst nicht geändert werden.

Überblick über die Werkzeuge der Debian-Betreuer enthält ein paar verschiedene Helferskripte. Das gängigste und beste Helfersystem (nach Meinung von Debian) ist `debhelper`. Frühere Helfersysteme wie `debmake` waren monolithisch. Man konnte nicht einen Teil des Helferskripts herausgreifen und verwenden, den man nützlich fand, sondern musste den Helfer für alles benutzen. `debhelper` besteht jedoch aus mehreren getrennten kleinen `dh_*`-Programmen. `dh_installman` installiert und komprimiert zum Beispiel Handbuchseiten, `dh_installmenu` installiert Menü-Dateien und so weiter. Daher bietet es eine ausreichende Flexibilität, die kleinen Helferskripte dort zu benutzen, wo sie nützlich sind, in Verbindung mit handgemachten Befehlen in `debian/rules`.

Sie können mit `debhelper` beginnen, indem Sie debhelper(7) lesen und sich die Beispiele ansehen, die dem Paket beigefügt sind. `dh_make` aus dem Paket `dh-make` (siehe *dh-make*) kann benutzt werden, um ein einfaches Quellpaket in ein mit `debhelper` bearbeitetes Paket umzuwandeln. Gleichwohl sollte diese Kurzform Sie jedoch nicht davon überzeugen, dass Sie sich nicht plagen müssen, um die einzelnen `dh_*`-Helfer zu verstehen. Falls Sie einen Helfer benutzen möchten, müssen Sie sich die Zeit nehmen zu lernen, wie dieser Helfer benutzt wird, um zu verstehen, was er erwartet und wie er sich verhält.

6.1.2 Pakete mit mehreren Binärdateien

Ein einzelnes Quellpaket erstellt häufig mehrere Binärpakete, entweder um mehrere Varianten derselben Software bereitzustellen (siehe z. B. das `vim`-Quellpaket) oder um mehrere kleine Pakete anstelle eines großem zu erstellen (z. B. damit der Benutzer sich nur die benötigte Teilmenge an Paketen installieren kann die der er benötigt, und dadurch auch Speicherplatz spart siehe z. B. das Quellpaket `libxml2`).

Der zweite Fall kann einfach in `debian/rules` verwaltet werden. Sie müssen nur die entsprechenden Dateien aus dem Build-Verzeichnis in die entsprechenden temporären Baumstrukturen des Pakets verschieben. Dies können Sie mit `install` oder `dh_install` aus `debhelper` erledigen. Sorgen Sie dafür, dass Sie die unterschiedlichen Umsetzungen der verschiedenen Pakete prüfen, um sicherzustellen, dass Sie die wechselseitigen Abhängigkeiten in `debian/control` richtig gesetzt haben.

Der erste Fall ist etwas schwieriger, da er mehrmaliges Neukompilieren der selben Software einschließt, aber mit unterschiedlichen Konfigurationsoptionen. Das `vim`-Quellpaket ist ein Beispiel dafür, wie dies mit einer handgemachten `debian/rules`-Datei gemacht wird.

6.2 Optimale Vorgehensweisen für `debian/control`

Die folgenden Vorgehensweisen sind maßgeblich für die Datei `debian/control`. Sie ergänzen die Richtlinien für Paketbeschreibungen.

Die Beschreibung des Pakets, wie es durch das entsprechende Feld in der Datei `control` definiert wird, enthält sowohl die Paketübersicht als auch die ausführliche Beschreibung des Pakets. *Allgemeine Leitlinien für Paketbeschreibungen* beschreibt übliche Richtlinien für beide Teile der Paketbeschreibung. Diesen folgend stellt *Die Paketübersicht oder Kurzbeschreibung* Richtlinien speziell für die Übersicht bereit und *Die ausführliche Beschreibung* enthält spezielle Richtlinien für die Beschreibung.

6.2.1 Der Paketname

Der Paketname:

- Muss mit weitreichend etablierten Konventionen übereinstimmen, z.B.
  - C- und C++-Bibliotheken enthalten typischerweise den Präfix lib
  - for many other languages, package names are prefixed or suffixed with the language name (the actual convention depends on the language).
- Sollte kein verbreitetes, ungeeignetes Wort sein. Insbesondere Wörter, die eine ganze Klasse von Applications repräsentieren (z.B. reader, browser) sollten für virtuelle Pakete reserviert sein.
- Sollte im Idealfall mindestens 4 Zeichen lang sein, außer der Upstream Name ist kürzer *und* das Projekt ist bereits weitgehend bekannt unter dem Namen (z.B. fzf).

6.2.2 Allgemeine Leitlinien für Paketbeschreibungen

Die Paketbeschreibung sollte für den erwarteten Durchschnittsanwender geschrieben werden, der Durchschnittsperson, die das Paket benutzt und davon profitiert. Entwicklungspakete sind beispielsweise für Entwickler und können in einer technischen Sprache verfasst werden. Anwendungen für allgemeinere Zwecke, wie Editoren, sollten für Anwender mit weniger technischem Verständnis geschrieben werden.

Die Durchsicht der Paketbeschreibungen mündet in der Schlussfolgerung, dass die meisten Paketbeschreibungen technischer Natur sind, also nicht so geschrieben, dass sie für Anwender ohne technischen Hintergrund einen Sinn ergeben. Sofern Ihr Paket nicht wirklich für technikkundige Anwender gedacht ist, ist dies ein Problem.

Wie können sie für nicht technikkundige Anwender schreiben? Vermeiden Sie Fachsprache. Vermeiden Sie, sich auf Anwendungen und Rahmenwerke zu beziehen, mit denen der Anwender möglicherweise nicht vertraut ist – GNOME oder KDE sind in Ordnung, da Anwender wahrscheinlich mit diesen Begriffen vertraut sind, aber GTK vermutlich nicht. Versuchen Sie nicht irgendein Wissen vorauszusetzen, geben Sie eine Einführung.

Seien Sie objektiv. Paketbeschreibungen sind nicht der richtige Ort, um Ihr Paket anzupreisen, egal wie sehr Sie es mögen. Denken Sie daran, dass der Leser nicht die gleichen Dinge als wichtig ansieht wie Sie.

Bezüge zu Namen anderer Softwarepakete, Protokollnamen, Standards oder Spezifikationen sollten, falls sie existiert, in ihrer vorschriftsmäßigen Form verwandt werden. Benutzen Sie zum Beispiel X Window System, X11 oder X, nicht X Windows, X-Windows, or X Window. Benutzen Sie GTK, nicht GTK+ oder gtk. Benutzen Sie GNOME, nicht Gnome. Benutzen Sie PostScript, nicht Postscript oder postscript.

Falls Sie Probleme beim Verfassen Ihrer Beschreibung haben, könnten Sie sie an `debian-l10n-english@lists.debian.org` senden und um Rückmeldung ersuchen.

6.2.3 Die Paketübersicht oder Kurzbeschreibung

Die Richtlinie legt fest, dass die Übersichtszeile (die Kurzbeschreibung) kurz sein muss, den Paketnamen nicht wiederholen darf, aber trotzdem informativ sein muss.

Die Übersichtsfunktionen sind ein Ausdruck, der das Paket beschreibt, kein kompletter Satz, daher ist Zeichensetzung unangebracht: Es wird weder eine besondere Großschreibung noch ein abschließender Punkt benötigt. Außerdem sollten jegliche bestimmten und unbestimmten Artikel am Anfang weggelassen werden – "a", "an" oder "the". Deshalb zum Beispiel:

```
Package: libeg0
Description: exemplification support library
```

Technisch gesehen ist dies eine Nominalphrase im Gegensatz zu einer Verbalphrase. Eine gute Entscheidungsregel ist, dass es möglich sein sollte, den Paket-*Namen* und die *Übersicht* in diesem Schema zu ersetzen:

Das Paket *Name* stellt {ein,eine,den,einige} *Übersicht* zur Verfügung.

Zusammenstellungen verwandter Pakete können ein alternatives Schema benutzen, das die Übersicht in zwei Teile unterteilt, als erstes eine Beschreibung der ganzen Suite und als zweites eine Zusammenfassung der Rolle, die das Paket darin spielt:

```
Package: eg-tools
Description: simple exemplification system (utilities)

Package: eg-doc
Description: simple exemplification system - documentation
```

Diese Übersicht folgt einem geänderten Schema. Wo ein Paket "*Name*" die Übersicht "*Suite* (*Rolle*)" oder "*Suite* - *Rolle*" hat, sollten die Elemente so ausgedrückt werden, dass sie in dieses Schema passen:

Das Paket *Name* stellt {ein,eine,die} *Rolle* für die *Suite* zur Verfügung.

6.2.4 Die ausführliche Beschreibung

Die ausführliche Beschreibung ist die wichtigste für den Benutzer verfügbare Information über ein Paket, bevor er es installiert. Sie sollte alle nötigen Informationen bereitstellen, damit der Anwender entscheiden kann, ob er das Paket installieren möchte. Es ist anzunehmen, dass der Benutzer bereits die Paketübersicht gelesen hat.

Die ausführliche Beschreibung sollte aus vollständigen Sätzen bestehen.

Der erste Absatz der ausführlichen Beschreibung sollte die folgenden Fragen beantworten: Was tut das Paket? Bei welchen Aufgaben hilft es dem Anwender? Es ist wichtig, dies auf eine nicht technische Weise zu beschreiben, sogar dann, wenn die Zielgruppe des Pakets notwendigerweise einen technischen Hintergrund hat.

Lange Beschreibungen verwandter Pakete, die beispielsweise aus derselben Quelle erstellt wurden, können Absätze gemeinsam nutzen, um die Konsistenz zu erhöhen und den Arbeitsaufwand für Übersetzer zu verringern. Sie benötigen jedoch mindestens einen separaten Absatz, der die spezifische Rolle des Pakets beschreibt.

Die folgenden Absätze sollten die folgenden Fragen beantworten: Warum benötige ich als Benutzer dieses Paket? Welche anderen Funktionen bietet das Paket? Welche herausragenden Funktionen und Mängel gibt es im Vergleich zu anderen Paketen (z.B. falls Sie X benötigen, benutzen Sie Y)? Steht das Paket in einem Zusammenhang mit anderen Paketen, die nicht vom Paketmanager gehandhabt werden (z.B. ist dies der Client für den Foo-Server)?

Seien Sie vorsichtig, um Rechtschreib- und Grammatikfehler zu vermeiden. Sorgen Sie für eine Rechtschreibprüfung. Sowohl `ispell` als auch `aspell` haben spezielle Modi zur Prüfung von `debian/control`-Dateien:

```
ispell -d american -g debian/control
```

```
aspell -d en -D -c debian/control
```

Anwender erwarten normalerweise, dass diese Fragen in der Paketbeschreibung beantwortet werden:

- Was tut das Paket? Falls es eine Erweiterung eines anderen Pakets ist, dann sollte eines Kurzbeschreibung des Pakets, das es erweitert, hier eingefügt werden.
- Warum sollte ich dieses Paket wollen? Dies bezieht sich auf das vorhergehende, aber nicht das gleiche (dies ist ein Mail-Client. Er ist toll, schnell, hat Schnittstellen zu PGP, LDAP und IMAP, hat die Funktionen X, Y und Z).
- Falls dieses Paket nicht direkt installiert werden sollte, sondern von einem anderen Paket mitinstalliert wird, sollte dies erwähnt werden.
- Falls das Paket `experimental` ist oder es andere Gründe gibt, weshalb es nicht benutzt werden sollte und wenn es andere Pakete gibt, die stattdessen benutzt werden sollen, sollte dies auch hier stehen.
- Was unterscheidet dieses Paket von der Konkurrenz? Ist es eine bessere Implementierung? Mehr Funktionen? Andere Funktionen? Warum sollte die Wahl auf dieses Paket fallen?

6.2.5 Homepage der Originalautoren

Es wird empfohlen, dass Sie die URL der Homepage des Pakets in das Feld `Homepage` im Abschnitt `Source` von `debian/control` hinzufügen. Diese Information in die Paketbeschreibung selbst einzutragen, wird als nicht erwünscht angesehen.

6.2.6 Ort des Versionsverwaltungssystems

Es gibt zusätzliche Felder für den Ort des Versionsverwaltungssystems in `debian/control`.

6.2.6.1 Vcs-Browser

Der Wert dieses Feldes sollte eine `https://`-URL sein, die auf eine via Webbrowser zu durchsuchende Kopie des Versionsverwaltungssystem-Repositorys verweist, das benutzt wird, um das angegebene Paket zu verwalten, falls verfügbar.

Die Information ist dazu gedacht, dem Endanwender zu helfen, diese Ressource aufzurufen um die letzte am Paket geleistete Arbeit durchstöbern zu können (z.B. wenn nach einem Patch gesucht wird, der einen Fehler behebt, der in der Fehlerdatenbank als `pending` gekennzeichnet ist).

6.2.6.2 Vcs-*

Der Wert dieses Feldes sollte eine Zeichenkette sein, die den Ort des Versionsverwaltungssystem-Repositorys eindeutig identifiziert, das zur Verwaltung des angegebenen Pakets benutzt wird, falls verfügbar. * identifiziert das Versionsverwaltungssystem. Aktuell werden vom Paketverfolgungssystem die folgenden Typen unterstützt: `arch`, `bzr` (Bazaar), `cvs`, `darcs`, `git`, `hg` (Mercurial), `mtn` (Monotone) und `svn` (Subversion).

Die Information ist für Benutzer bestimmt, die im gegebenen Versionsverwaltungssystem sachkundig sind und bereit, die aktuelle Version des Pakets aus den VCS-Quellen zu erstellen. Andere Nutzungen dieser Information könnten das automatische Erstellen der letzten VCS-Version des Pakets umfassen. Dazu sollte der vom Feld angegebene Ort versionsunabhängig sein und auf den Hauptzweig (main branch) zeigen (bei Versionsverwaltungssystemen, die dieses Konzept unterstützen). Außerdem sollte der Endanwender auf den Ort zugreifen können, auf

den verwiesen wird. Die Erfüllung dieser Anforderungen könnte einen anonymen Zugriff voraussetzen, statt auf eine Version mit SSH-Zugriff zu verweisen.

Im folgendem Beispiel wird eine Instanz des Felds für ein Git-Repository des `vim`-Pakets angezeigt. Beachten Sie, dass die URL das `https://`- Schema (anstelle von `ssh://`) benutzt. Die Verwendung der Felder `Vcs-Browser` und `Homepage` wird ebenfalls gezeigt, wie oben beschrieben.

```
Source: vim
<snip>
Vcs-Git: https://salsa.debian.org/vim-team/vim.git
Vcs-Browser: https://salsa.debian.org/vim-team/vim
Homepage: https://www.vim.org
```

Das Verwalten der Paketierung in einem Versionskontrollsystem und das Festlegen eines Vcs-*-Headers ist eine bewährte Vorgehensweise und erleichtert anderen das Einbringen von Änderungen.

Fast alle Pakete in Debian, die ein Versionskontrollsystem verwenden, verwenden Git. Wenn Sie ein neues Paket erstellen, ist die Verwendung von Git eine gute Idee, einfach weil es das System ist, mit dem die Mitwirkenden vertraut sind.

DEP-14 definiert ein allgemein gültiges Layout für Debian Pakete.

6.3 Optimale Vorgehensweisen für `debian/changelog`

Die folgenden Vorgehensweisen ergänzen die Richtlinien für Änderungsprotokolldateien.

6.3.1 Verfassen nützlicher Änderungsprotokolleinträge

Der Änderungsprotokolleintrag (Changelog) einer Paketüberarbeitung dokumentiert Änderungen der aktuellen Version, und nur diese. Der Schwerpunkt liegt auf der Beschreibung bedeutender und für den Anwender sichtbarer Änderungen, die seit der letzten Version vorgenommen wurden.

Der Fokus liegt darauf, *was* geändert wurde – wer, wie und wann ist normalerweise nicht so wichtig. Erinnern Sie gleichwohl höflich an die Leute, die merklich Hilfe beim Erstellen des Pakets geleistet haben (die z.B. Patches gesandt haben).

Es ist nicht nötig, belanglose und offensichtliche Änderungen näher auszuführen. Sie können außerdem mehrere Änderungen in einem Eintrag zusammenfassen. Fassen Sie sich andererseits nicht zu kurz, falls Sie eine größere Änderung vorgenommen haben. Stellen Sie insbesondere klar, falls es Änderungen gibt, die das Verhalten des Programms ändern. Benutzen Sie für weitere Erklärungen die Datei `README.Debian`.

Benutzen Sie geläufiges Englisch, so dass die Mehrheit der Leser es begreifen kann. Vermeiden Sie Abkürzungen, technische Begriffe und Fachsprache, wenn Sie Änderungen erklären, die Fehlerberichte schließen, insbesondere bei Fehlern die von Anwendern eingereicht wurden, die Ihnen als technisch unerfahren aufgefallen sind. Seien Sie höflich, fluchen Sie nicht.

Manchmal ist es wünschenswert, den Änderungsprotokolleinträgen die Namen der Dateien voranstellen, die geändert wurden. Es ist jedoch nicht nötig, explizit jede einzelne geänderte Datei aufzuführen, insbesondere dann nicht, wenn die Änderung klein oder wiederholend war. Sie können Platzhalter verwenden.

Treffen Sie keine Annahmen wenn Sie sich auf Fehler beziehen. Sagen Sie welches Problem vorlag, wie es behoben wurde und hängen Sie die Zeichenkette "closes: #nnnnn" an. Weitere Informationen erhalten Sie unter *Wann Fehler durch neue Uploads geschlossen werden*.

6.3.2 Auswahl Dringlichkeit des Uploads

Das Release-Team hat angegeben, dass sie erwarten, dass die meisten Uploads nach `unstable` die Dringlichkeit **urgency=medium** verwenden. Deshalb sollten Sie **urgency=medium** benutzen, es sei denn, es gibt spezielle Gründe, dass der Upload schneller oder langsamer nach `testing` migrieren soll (Siehe auch:ref:*testing-unstable*). Beispielsweise können Sie **urgency=low** nehmen, falls die Änderungen seit dem letzten Upload sehr umfangreiche oder disruptive, unerwartete Änderungen enthalten.

The delays are currently 2, 5 or 10 days, depending on the urgency (high, medium or low). The actual numbers are actually controlled by the britney configuration which also includes accelerated migrations when Autopkgtest passes.

6.3.3 Häufige Missverständnisse über Änderungsprotokolleinträge

Die Änderungsprotokolleinträge sollten **keine** allgemeinen Paketierungsthemen dokumentieren (Hey, falls Sie die Foo.conf suchen, die ist in /etc/blah/.), da von Administratoren und Anwendern angenommen wird, dass sie zumindest entfernt damit vertraut sind, wie solche Dinge im Allgemeinen auf Debian-Systemen eingerichtet sind. Erwähnen Sie jedoch, wenn Sie den Ort einer Konfigurationsdatei ändern.

Die einzigen Fehler, die mit einem Änderungsprotokolleintrag geschlossen werden, sollten Fehler sein, die tatsächlich in der gleichen Überarbeitung des Pakets behoben werden. Das Schließen von Fehlern ohne Bezug dazu, ist eine falsche Vorgehensweise. Siehe *Wann Fehler durch neue Uploads geschlossen werden*.

Die Änderungsprotokolleinträge sollten **nicht** für zufällige Diskussionen mit Leuten, die Fehler melden (Ich kann keine Schutzverletzungen sehen, wenn ich Foo mit der Option Bar starte. Senden Sie weitere Informationen.), allgemeine Äußerungen über das Leben, das Universum und alles mögliche (Entschuldigung, dass der Upload so lange brauchte, aber ich hatte die Grippe.) oder Hilfeersuchen (Die Fehlerliste für dieses Paket ist riesig, bitte packen Sie mit an) benutzt werden. Solche Dinge werden normalerweise nicht von Ihrer Zielgruppe bemerkt, könnten aber viele Leute stören, die Informationen über tatsächliche Änderungen am Paket lesen möchten. Weitere Informationen über die Benutzung der Fehlerdatenbank finden Sie unter *Auf Fehler antworten*.

Es ist ein alter Brauch, im ersten regulären Upload des Paketbetreuers das Beheben von Fehlern durch Non-Maintainer-Uploads zu bestätigen. Da Debian nun über eine Versionsverwaltung verfügt, reicht es aus, die NMU-Änderungsprotokolleinträge zu erhalten und diese Tatsache nur in Ihrem eigenen Änderungsprotokolleintrag zu erwähnen.

6.3.4 Häufige Fehler in Änderungsprotokolleinträgen

Die folgenden Beispiele demonstrieren einige häufige Fehler oder Beispiele für schlechten Stil in Änderungsprotokolleinträgen.

```
* Fixed all outstanding bugs.
```

Dies teilt den Lesern offensichtlich nichts Nützliches mit.

```
* Applied patch from Jane Random.
```

Was war das für ein Patch?

```
* Late night install target overhaul.
```

Welche Korrektur wurde ausgeführt? Soll die Erwähnung der späten Nacht daran erinnern, dass man dem Code nicht trauen sollte?

```
* Fix vsync fw glitch w/ ancient CRTs.
```

Hier werden zu viele Akronyme benutzt! Was bedeuted "fw", eventuell "firmware"? Und es ist nicht allzu klar, worum es bei dem Fehler tatsächlich ging oder wie er behoben wurde.

```
* This is not a bug, closes: #nnnnnn.
```

Erst einmal ist es absolut unnötig, das Paket hochzuladen, um diese Information zu übermitteln. Benutzen Sie stattdessen die Fehlerdatenbank. Zweitens fehlt die Erklärung, warum dieser Bericht kein Fehler ist.

```
* Has been fixed for ages, but I forgot to close; closes: #54321.
```

Falls Sie aus irgend einem Grund die Fehlernummer in einem früheren Änderungsprotokolleintrag nicht erwähnt haben, ist das kein Problem. Schließen Sie den Fehler einfach im BTS. Es ist nicht nötig, die Änderungsprotokolldatei anzufassen, vorausgesetzt, die Beschreibung der Fehlerbehebung ist bereits darin enthalten (dies gilt auch für

Korrekturen durch die Originalautoren/-Betreuer. Sie müssen keine Fehler verfolgen, die diese bereits vor Jahren in Ihrem Änderungsprotokoll behoben haben).

```
* Closes: #12345, #12346, #15432
```

Wo ist die Beschreibung? Falls Ihnen keine aussagekräftige Nachricht einfällt, beginnen Sie damit, die Titel der verschiedenen Fehler einzufügen.

6.3.5 Änderungsprotokolle mit `NEWS.Debian`-Dateien ergänzen

Wichtige Nachrichten über Änderungen in einem Paket können auch in die `NEWS.Debian`-Dateien geschrieben werden. Die Nachrichten werden durch Werkzeuge wie `apt-listchanges` vor dem ganzen Rest des Änderungsprotokolls angezeigt. Dies ist das bevorzugte Mittel, dem Anwender bedeutende Änderungen in einem Paket mitzuteilen. Es ist besser, als `debconf`-Notizen zu benutzen, da es weniger stört und der Anwender nach der Installation zurückgehen und in der `NEWS.Debian`-Datei nachschlagen kann. Es ist auch besser, als die Hauptänderungen in `README.Debian` aufzuführen, da der Anwender solche Notizen leicht übersehen kann.

Das Dateiformat entspricht dem des Änderungsprotokolls, die Sternchen werden allerdings weggelassen und jedes Nachrichtenelement wird, wenn nötig, mit einem vollständigen Satz beschrieben, statt der kurz gefassten Zusammenfassungen, die in ein Änderungsprotokoll einfließen. Sie sind gut beraten, Ihre Datei durch `dpkg-parsechangelog` laufen zu lassen, um die Formatierung zu prüfen, da sie nicht automatisch während des Builds getestet wird, so wie dies beim Änderungsprotokoll geschieht. Hier nun ein Beispiel einer echten `NEWS.Debian`-Datei:

```
cron (3.0pl1-74) unstable; urgency=low

    The checksecurity script is no longer included with the cron package:
    it now has its own package, checksecurity. If you liked the
    functionality provided with that script, please install the new
    package.

 -- Steve Greenland <stevegr@debian.org>  Sat,  6 Sep 2003 17:15:03 -0500
```

Die Datei `NEWS.Debian` wird als `/usr/share/doc/`*Paket*`/NEWS.Debian.gz` installiert. Sie ist komprimiert und hat immer diesen Namen, auch in nativen Debian-Paketen. Falls Sie `debhelper` benutzen, wird `dh_installchangelogs` die `debian/NEWS`-Dateien für Sie installieren.

Anders als Änderungsprotokolldateien müssen Sie die `debian/NEWS`-Dateien nicht bei jeder Veröffentlichung aktualisieren. Aktualisieren Sie diese nur, wenn Sie etwas besonders berichtenswertes haben, worüber der Anwender Bescheid wissen sollte. Falls Sie überhaupt keine Nachrichten haben, ist es nicht nötig, Ihrem Paket eine `debian/NEWS`-Datei mitzugeben. Keine Nachricht ist eine gute Nachricht!

6.4 Optimale Vorgehensweisen rund um Sicherheit

When an upstream publishes a cryptographic signature for every new release, you should setup `uscan` to automatically verify the latter. For details, refer to the section on `Upstream source location: "debian/watch"` in the Debian Policy Manual.

https://wiki.debian.org/Hardening enthält Vorschläge wie ein ausführbares Programm gebaut werden kann, so dass es in Bezug auf Sicherheit gehärtet ist.

6.5 Optimale Vorgehensweisen für Betreuerskripte

Zu den Betreuerskripten gehören die Dateien `debian/postinst`, `debian/preinst`, `debian/prerm` und `debian/postrm`. Diese Skripte kümmern sich um die Paketeinrichtung bei jeder Installation oder Deinstallation des Pakets, bei dem es nicht ausreicht, nur die Dateien und Verzeichnisse zu erstellen oder zu entfernen. Die folgenden Anweisungen ergänzen die Debian Policy.

6.5.1 Generelle Richtlinien

Betreuerskripte müssen idempotent sein. Das bedeutet, dass Sie sicherstellen müssen, dass nichts Schlimmes passiert, wenn das Skript zweimal aufgerufen wird, während es normalerweise nur einmal aufgerufen würde.

Standardein- und -ausgabe könnten zu Protokollierungszwecken umgeleitet werden (z.B. in Pipes), verlassen Sie sich daher nicht darauf, dass dies auf einem Terminal ausgegeben werden.

Jegliche Bedienerführung oder interaktive Konfiguration sollte so gering wie möglich gehalten werden. Wenn es nötig ist, sollten Sie das Paket `debconf` für die Schnittstelle benutzen. Denken Sie daran, dass diese Bedienerführung nur in der `configure`-Stufe des `postinst`-Skripts stattfinden kann.

Halten Sie die Betreuerskripte so einfach wie möglich. Es wird empfohlen, nur reine POSIX-Shell-Skripte zu benutzen. Falls Sie irgendwelche Bash-Funktionen benötigen, vergessen Sie nicht, dass das Betreuerskript eine Shebang-Zeile haben muss. POSIX-Shell oder Bash werden für Perl bevorzugt, da sie `debhelper` ermöglichen, den Skripten einfach Teile hinzuzufügen.

Falls Sie Ihre Betreuerskripte ändern, stellen Sie sicher, dass Sie das Entfernen des Pakets, die mehrmalige Installation und das vollständige Entfernen testen. Vergewissern Sie sich, dass nach dem vollständigen Entfernen des Pakets alles komplett gelöscht ist, sprich es muss jede Datei entfernt sein, die direkt oder indirekt in irgendeinem Betreuerskript erstellt wurde.

Falls Sie prüfen möchten, ob ein Befehl existiert, sollten Sie z.B. so etwas benutzen

```
if command -v install-docs > /dev/null; then ...
```

Sie können diese Funktion benutzen, um `$PATH` nach einem Befehlsnamen zu durchsuchen, der als Argument übergeben wird. Sie gibt "true" (null) zurück, falls der Befehl gefunden wurde und "false" falls nicht. Dies ist wirklich die am ehesten portierbare Möglichkeit, da `command -v` ein Shell-Builtin für viele Shells ist und im POSIX-Standard definiert ist.

Das Benutzen von `which` ist eine akzeptable Alternative da es im erforderlichen Paket `debianutils` enthalten ist.

6.5.2 Erwartetes Verhalten von komplexen postinst Skripten

`postinst` Skripte sollten normalerweise nicht mit einem Fehler beendet werden, außer in sehr seltenen Fällen. In diesen Fällen muss ein postinst Skript erfolgreich beendet werden:

bei neuen Installationen:

- die Konfiguration des Service ist einfach und es kann angenommen werden, dass sie in typischen Debian-Setups ohne Änderungen funktioniert
- der Service hat keine externen Abhängigkeiten (z.B. eine Datenbank, welche noch nicht konfiguriert oder bei der Installation unerreichbar ist)

bei Upgrades:

- `postinst` kann mit hoher Sicherheit (welche nicht immer möglich sein mag) verifizieren, dass der Service vor dem Neustart lief
- der Service hat keine externen Abhängigkeiten oder `postinst` kann verifizieren, dass diese funktionieren
- die Konfiguration des Service hat sich zwischen der alten und der neuen Paket-Version nicht derart geändert, dass die Änderung inkompatibel mit der alten Version ist

6.6 Konfigurationsverwaltung mit `debconf`

`Debconf` ist ein Konfigurationsverwaltungssystem, das von allerlei Paketierungsskripten (hauptsächlich `postinst`) benutzt werden kann, um Rückmeldungen von Anwendern betreffend der Paketkonfiguration abzufragen. Direkte Benutzer-Interaktion muss nun zugunsten der Interaktion mit `debconf` vermieden werden, damit in Zukunft nicht interaktive Installationen möglich sind.

Debconf ist ein großartiges Werkzeug, aber es wird oft mangelhaft benutzt. Viele alltägliche Fehler sind auf der Handbuchseite debconf-devel(7) aufgeführt. Sie sollten sie lesen, falls Sie sich entscheiden debconf zu benutzen. Außerdem werden hier ein paar optimale Vorgehensweisen vorgestellt.

Diese Leitlinien enthalten einige Empfehlungen für Schreibstil und Typografie, allgemeine Betrachtungen über die Benutzung von debconf sowie spezifischere Empfehlungen für einige Teile der Distribution (das Installationssystem beispielsweise).

6.6.1 Missbrauchen Sie debconf nicht

Seit debconf in Debian erschien, wurde es öfter missbraucht und viel von der Kritik, die bei der Debian-Distribution einging, rührte vom debconf-Missbrauch her und bemängelte die Notwendigkeit, ein großes Fragenbündel beantworten zu müssen, bevor eine Kleinigkeit installiert war.

Lassen Sie Hinweise zur Benutzung der Software dort wozu sie gehören: In den Dateien `NEWS.Debian` oder `README.Debian`. Benutzen Sie debconf-Meldungen nur für wichtige Anmerkungen die direkt die Benutzbarkeit des Pakets betreffen. Bedenken Sie, dass diese die Installation immer blockieren bis diese bestätigt wurden, oder Sie belästigen den Anwender per E-Mail.

Wählen Sie die Prioritäten der Fragen in Betreuerskripten sorgfältig. Einzelheiten über Prioritäten finden Sie unter debconf-devel 7. Die meisten Fragen sollten mittlere oder niedrige Prioritäten nutzen.

6.6.2 Allgemeine Empfehlungen für Autoren und Übersetzer

6.6.2.1 Schreiben Sie korrektes Englisch

Die meisten Debian-Paketbetreuer haben nicht Englisch als Muttersprache. Daher ist es für sie nicht einfach, korrekt formulierte Vorlagen zu verfassen.

Bitte benutzen (und missbrauchen) Sie die Mailingliste `debian-l10n-english@lists.debian.org`. Lassen Sie Ihre Vorlagen dort Korrekturlesen.

Schlecht geschriebene Vorlagen werfen ein armseliges Bild auf Ihr Paket, Ihre Arbeit ... oder sogar auf Debian selbst.

Vermeiden Sie soweit möglich technische Fachsprache. Auch wenn sich einige Begriffe für Sie vertraut anhören, könnten sie für andere unverständlich sein. Falls sie sich nicht vermeiden lassen, versuchen Sie diese zu erklären (benutzen Sie die erweiterte Beschreibung). Versuchen Sie dabei, zwischen Aussagekraft und Einfachheit abzuwägen.

6.6.2.2 Seien Sie freundlich zu Übersetzern

Debconf-Vorlagen können übersetzt werden. Debconf bietet zusammen mit seinem Schwesterpaket `po-debconf` ein einfaches Werkzeug, um Vorlagen durch Übersetzer-Teams oder sogar einzelne Personen übersetzen zu lassen.

Bitte benutzen Sie Gettext-basierte Vorlagen. Installieren Sie `po-debconf` auf Ihrem Entwicklungssystem und lesen Sie dessen Dokumentation (`man po-debconf` ist ein guter Anfang).

Vermeiden Sie es, Vorlagen häufig zu ändern. Das Ändern von Vorlagen führt zu Mehrarbeit für Übersetzer, deren Übersetzungen unvollständig werden. Eine unvollständige Übersetzung ist eine Zeichenkette, bei der sich seit dem Übersetzen das Original geändert hat und das daher eine Aktualisierung durch den Übersetzer benötigt. Wenn die Änderungen klein genug sind, wird die Originalübersetzung in den PO-Dateien beibehalten, jedoch mit `fuzzy` gekennzeichnet.

Falls Sie planen, Änderungen an Ihren Originalvorlagen vorzunehmen, benutzen Sie bitte das Benachrichtigungssystem namens `podebconf-report-po`, das vom Paket `po-debconf` bereitgestellt wird, um die Übersetzer zu kontaktieren. Die meisten aktiven Übersetzer sind sehr zugänglich, und deren Arbeit zusammen mit Ihren geänderten Vorlagen einzubeziehen, wird Sie vor zusätzlichen Uploads bewahren. Falls Sie Gettext-basierte Vorlagen verwenden, werden die Namen und E-Mail-Adressen der Übersetzer in den Kopfzeilen der PO-Dateien erwähnt und von `podebconf-report-po` benutzt.

Ein empfohlene Art, das Hilfswerkzeug zu benutzen ist:

```
cd debian/po && podebconf-report-po --call --languageteam --withtranslators --
↪deadline="+10 days"
```

Dieser Befehl wird zuerst die PO- und POT-Dateien in `debian/po` mit den in `debian/po/POTFILES.in` aufgeführten Vorlagendateien synchronisieren. Dann wird er einen Aufruf für neue Übersetzungen an die Mailingliste `debian-i18n@lists.debian.org` senden. Am Schluss wird er außerdem einen Aufruf für neue Übersetzungen an das Sprach-Team (im Feld `Language-Team` jeder PO-Datei erwähnt), sowie den letzten Übersetzer (aus `Last-Translator`) senden.

Es wird immer gewürdigt, wenn Sie den Übersetzern einen Abgabetermin nennen, so dass diese ihre Arbeit organisieren können. Bitte denken Sie daran, dass einige Übersetzer-Teams einen formalisierten Übersetzungs-/Korrekturprozess haben und eine Zeitspanne, die kürzer als zehn Tage ist, als unangemessen angesehen wird. Eine kürzere Frist übt zuviel Druck auf die Übersetzer-Teams aus und sollte nur für sehr kleine Änderungen gewählt werden.

Im Zweifelsfall können Sie auch das Übersetzer-Team für eine bestimmte Sprache (debian-l10n-xxxxx@lists.debian.org) oder die Mailingliste `debian-i18n@lists.debian.org` kontaktieren.

6.6.2.3 Entfernen Sie die fuzzy-Markierungen in vollständigen Übersetzungen, wenn Sie Tipp- und Rechtschreibfehler korrigieren

Wenn der Text einer Debconf-Vorlage korrigiert wurde und Sie **sicher** sind, dass die Änderung **keine** Übersetzungen beeinflusst, seien Sie so nett zu den Übersetzern, die *fuzzy*-Markierungen aus deren Übersetzungen zu entfernen.

Falls Sie dies nicht tun, wird die ganze Vorlage nicht übersetzt sein, bis Ihnen ein Übersetzer eine Aktualisierung zusendet.

Um die *fuzzy*-Markierungen aus Übersetzungen zu entfernen, können Sie `msguntypot` benutzen (Teil des Pakets `po4a`).

1. Erzeugen Sie die POT- und PO-Dateien neu.

   ```
   debconf-updatepo
   ```

2. Erstellen Sie eine Kopie der POT-Datei.

   ```
   cp templates.pot templates.pot.orig
   ```

3. Erstellen Sie eine Kopie aller PO-Dateien.

   ```
   mkdir po_fridge; cp *.po po_fridge
   ```

4. Ändern Sie die Debconf-Vorlagedateien, um den Tippfehler zu korrigieren.

5. Erzeugen Sie die POT- und PO-Dateien (wieder) neu.

   ```
   debconf-updatepo
   ```

   An dieser Stelle markiert die Korrektur des Tippfehlers alle Übersetzungen mit "fuzzy" und diese unglückliche Änderung ist die einzige zwischen den PO-Dateien Ihres Hauptverzeichnisses und denen aus fridge. Hier nun eine Erklärung, wie das gelöst wird.

6. Verwerfen Sie die mit "fuzzy" markierte Übersetzung und stellen Sie die aus fridge wieder her.

   ```
   cp po_fridge/*.po .
   ```

7. Führen Sie manuell die PO-Dateien mit der neuen POT-Datei zusammen, unter Berücksichtigung/Vermeidung des nutzlosen "fuzzy".

   ```
   msguntypot -o templates.pot.orig -n templates.pot *.po
   ```

8. Räumen Sie auf.

```
rm -rf templates.pot.orig po_fridge
```

6.6.2.4 Treffen Sie keine Annahmen über Schnittstellen

Vorlagentexte sollten keinen Bezug zu Steuerelementen herstellen, die zu irgendwelchen Debconf-Schnittstellen gehören. Sätze wie *Wenn Sie mit Ja antworten...* haben keinen Sinn für Benutzer grafischer Oberflächen, weil dort bei logischen Fragen Kontrollkästchen angehakt werden.

Zeichenkettenvorlagen sollten außerdem vermeiden, Vorgabewerte in ihrer Beschreibung zu erwähnen. Erstens sind diese zusätzlich zu den Werten, die der Anwender sieht, vorhanden. Außerdem könnten sich diese Werte von der Auswahl des Paketbetreuers unterscheiden (zum Beispiel, wenn die Debconf-Datenbank voreingestellt ist).

Versuchen Sie, allgemein ausgedrückt, Bezug auf Benutzeraktionen zu vermeiden. Geben Sie nur Tatsachen wieder.

6.6.2.5 Reden Sie nicht in der ersten Person

Sie sollten vermeiden, in der ersten Person zu reden (*Ich werde das tun...* oder *Wir empfehlen...*). Der Rechner ist keine Person und die Debconf-Vorlagen sprechen nicht stellvertretend für Debian-Entwickler. Sie sollten neutrale Formulierungen benutzen. Diejenigen unter Ihnen, die bereits wissenschaftliche Publikationen verfasst haben, können ihre Vorlagen so schreiben, als würden Sie wissenschaftliche Papiere verfassen. Versuchen Sie jedoch, wenn möglich eine aktive Anrede zu verwenden, wie *Aktivieren Sie dies, wenn ...* anstelle von *Dies kann aktiviert werden, wenn...*.

6.6.2.6 Formulieren Sie geschlechtsneutral

Um unser Bekenntnis zu unserem Diversity Statement zu zeigen, verwenden Sie bitte in Ihren Texten geschlechtergerechte Formulierungen, indem Sie Pronomen wie "Er" oder "Sie" vermeiden, wenn Sie auf eine Aufgabenrolle (wie zum Beispiel Betreuer) Bezug nehmen.

6.6.3 Definition von Vorlagenfeldern

Dieser Teil stellt einige Informationen bereit, die überwiegend von der Handbuchseite debconf-devel(7) übernommen wurden.

6.6.3.1 Typen

string

Resultiert in einem Eingabefeld freier Form, in das der Benutzer jegliche Zeichenkette eingeben kann.

password

Gibt dem Benutzer eine Eingabeaufforderung für ein Passwort aus. Benutzen Sie dies mit Vorsicht. Vergegenwärtigen Sie sich, dass das Passwort das der Benutzer eingibt in die Debconf-Datenbank geschrieben wird. Sie sollten diesen Wert möglicherweise aus der Datenbank löschen, sobald dies möglich ist.

boolean

Eine Auswahl "wahr/falsch". Denken Sie daran: true/false, nicht **yes/no** ...

select

Eine Auswahl aus mehreren Werten. Die Auswahlmöglichkeiten müssen in einem "Choices" benannten Feld angegeben werden. Trennen Sie die möglichen Werte mit Komma und Leerzeichen, wie hier: `Choices: yes, no, maybe`.

Falls Auswahlmöglichkeiten übersetzbare Zeichenketten sind, kann das Feld durch Benutzung von `__Choices` als übersetzbar gekennzeichnet werden. Der doppelte Unterstrich wird jede Auswahl in eine separate Zeichenkette heraus trennen.

Das System po-debconf bietet außerdem interessante Möglichkeiten, um nur **einige** Auswahlmöglichkeiten als übersetzbar zu kennzeichnen. Ein Beispiel:

```
Template: foo/bar
Type: Select
#flag:translate:3
__Choices: PAL, SECAM, Other
_Description: TV standard:
 Please choose the TV standard used in your country.
```

In diesem Beispiel ist nur die Zeichenkette "Other" übersetzbar, während das andere Abkürzungen sind, die nicht übersetzt werden sollten. Obiges ermöglicht, dass nur "Other" in die POT- und PO-Dateien eingefügt wird.

Das Schaltersystem der Debconf-Vorlagen bietet viele solcher Möglichkeiten. Die Handbuchseite po-debconf(7) führt all diese Möglichkeiten auf.

multiselect

Like the select data type, except the user can choose any number of items from the choices list (or choose none of them).

note

Statt per se als Frage stellt dieser Datentyp einen Hinweis dar, der dem Benutzer angezeigt werden kann. Er sollte nur für wichtige Anmerkungen benutzt werden, die der Benutzer wirklich sehen sollte, weil debconf großen Aufwand betreibt, um sicherzustellen, dass der Benutzer sie wahrnimmt; die Installation wird gestoppt, bis der Benutzer eine Taste drückt, in manchen Fällen bekommt er sogar eine Benachrichtigung per E-Mail.

text

Dieser Typ wird nun als veraltet angesehen: Benutzen Sie ihn nicht.

error

Dieser Typ wurde für Fehlermeldungen entworfen. Er ist dem Typ "note" am ähnlichsten. Oberflächen könnten ihn unterschiedlich anzeigen (die Oberfläche von Cdebconf zeichnet beispielsweise einen roten statt des üblichen blauen Bildschirms).

Es wird empfohlen, diesen Typ für jegliche Nachricht zu verwenden, die die Aufmerksamkeit des Anwenders für irgendeine Art von Korrektur auf sich ziehen muss.

6.6.3.2 Description: Kurze und erweiterte Beschreibung

Vorlagenbeschreibungen haben zwei Teile: kurz und erweitert. Die Kurzbeschreibung steht in der Zeile "Description:" der Vorlage.

Die Kurzbeschreibung sollte knapp gehalten werden (ungefähr 50 Zeichen), so dass sie in den meisten debconf-Schnittstellen untergebracht werden kann. Es hilft obendrein Übersetzern, wenn sie kurz gehalten wird, da Übersetzungen normalerweise dazu neigen, länger als das Original zu sein.

Die Kurzbeschreibung sollte für sich allein stehen können. Einige Schnittstellen zeigen die ausführliche Beschreibung standardmäßig nicht an, oder nur, wenn der Benutzer explizit danach fragt, oder eventuell wird sie gar nicht angezeigt. Vermeiden Sie so was wie "Was möchten Sie tun?"

Die Kurzbeschreibung muss nicht notwendigerweise aus einem vollständigen Satz bestehen. Dies ist Teil der Forderung nach kurzen, brauchbaren Empfehlungen.

Die erweiterte Beschreibung sollte die Kurzbeschreibung nicht Wort für Wort wiederholen. Falls Ihnen keine ausführliche Beschreibung einfällt, denken Sie zuerst etwas darüber nach. Schreiben Sie an debian-devel. Bitten Sie um Hilfe. Nehmen Sie Schreibunterricht! Diese erweiterte Beschreibung ist wichtig. Falls Sie nach allem noch immer nicht damit zurecht kommen, lassen Sie sie leer.

Die erweiterte Beschreibung sollte in ganzen Sätzen verfasst sein. Absätze sollten kurz gehalten werden, um die Leserlichkeit zu verbessern. Vermischen Sie nicht zwei Ideen in einem Absatz, sondern benutzen Sie lieber einen anderen Absatz.

Don't be too verbose. Users tend to ignore too long screens. 20 lines are by experience a border you shouldn't cross, because that means that in the classical dialog interface, people will need to scroll, and lot of people just don't do that.

Die erweiterte Beschreibung sollte **keine** Frage enthalten.

Um etwas über besondere Regeln zu erfahren, die vom Vorlagentyp (string, boolean etc.) abhängen, lesen Sie das Folgende.

6.6.3.3 Choices

Dieses Feld sollte für "select"- und "multiselect"-Typen verwandt werden. Es enthält die Auswahlmöglichkeiten, die dem Benutzer angezeigt werden. Die Auswahlmöglichkeiten sollten durch Komma getrennt werden.

6.6.3.4 Default

Dieses Feld ist optional. Es enthält die vorausgewählte Antwort für die "string"-, "select"- und "multiselect"-Vorlagen. Für "multiselect"-Vorlagen könnte es eine durch Komma getrennte Auswahlliste enthalten.

6.6.4 Gestaltungsrichtlinie für Vorlagenfelder

6.6.4.1 Feld "Type"

Keine besondere Angabe außer: Benutzen Sie den geeigneten Typ bezogen auf den vorhergehenden Abschnitt.

6.6.4.2 Feld "Description"

Es folgen spezifische Anweisungen für ordnungsgemäßes Verfassen der Beschreibung (kurz und erweitert), abhängig vom Vorlagentyp.

"string"-/"password"-Vorlagen

- Die Kurzbeschreibung ist eine Abfrage und **kein** Titel. Vermeiden Sie den Fragestil (IP-Adresse?) und geben Sie offenen Abfragen (IP-Adresse:) den Vorzug. Es wird empfohlen, Doppelpunkte zu benutzen.
- Die erweiterte Beschreibung ist eine Ergänzung der Kurzbeschreibung. Im erweiterten Teil erklären Sie, was gefragt ist, anstatt die gleiche Frage in längerer Formulierung wieder zu stellen. Benutzen Sie ganze Sätze. Von knappem Schreibstil wird strikt abgeraten.

"boolean"-Vorlagen

- Die Kurzbeschreibung sollte als Frage formuliert und kurz gehalten werden, und generell mit einem Fragezeichen enden. Kompakter Schreibstil ist erlaubt und sogar gewollt, falls die Frage eher lang ist. (Denken Sie daran, dass Übersetzungen oft länger als die Originalversionen sind.)
- Nochmals: Bitte vermeiden Sie, sich auf Schnittstellen-spezifische Dinge zu beziehen. Ein häufiger Fehler bei solchen Vorlagen ist, auf Ja-Typ-Konstruktionen zu antworten.

"select"/"multiselect"

- Die Kurzbeschreibung ist eine Abfrage und **kein** Titel. Benutzen Sie **keine** nutzlosen "Bitte wählen Sie..."-Konstruktionen. Anwender sind klug genug, um herauszufinden, dass sie etwas auswählen sollen ... :)
- Die erweiterte Beschreibung wird die Kurzbeschreibung vervollständigen. Sie könnte sich auf die verfügbaren Auswahlmöglichkeiten beziehen. Sie könnte zudem erwähnen, dass der Anwender unter mehr als einer verfügbaren Auswahlmöglichkeit wählen kann, falls es sich um eine "multiselect"-Vorlage handelt.

"notes"

- Die Kurzbeschreibung sollte als **Titel** betrachtet werden.
- Die erweiterte Beschreibung ist das, was als detaillierte Erklärung der Notiz angezeigt wird. Sätze, kein knapper Schreibstil.
- **Missbrauchen Sie debconf nicht.** Anmerkungen sind die häufigste Art, auf die debconf missbraucht wird. Wie steht doch in der Handbuchseite von debconf-devel(7) geschrieben: Es ist am Besten, dies nur für Warnungen über sehr ernsthafte Probleme zu benutzen. Die Dateien `NEWS.Debian` oder `README.Debian` sind für viele Notizen der passende Ort. Wenn Sie dies lesen, und darüber nachdenken, Ihre "notes" zu Einträgen in `NEWS.Debian` oder `README.Debian` umzuwandeln, bewahren Sie bitte dennoch Ihre existierenden Übersetzungen für die Zukunft auf.

6.6.4.3 Das Feld "Choices"

Falls sich "Choices" zu oft ändert, sollten Sie in Betracht ziehen, zum __Choices-Trick zu greifen. Dies wird jede einzelne Auswahl in eine einzelne Zeichenkette aufteilen, was Übersetzern beträchtlich bei ihrer Arbeit helfen wird.

6.6.4.4 Das Feld "Default"

Wenn der Standardwert für eine ausgewählte Vorlage wahrscheinlich von der Benutzersprache abhängt (z. B. wenn es sich bei der Auswahl um eine Sprachauswahl handelt), verwenden Sie bitte den in po-debconf 7 dokumentierten Trick _Default.

Dieses Spezialfeld ermöglicht Übersetzern, die am Besten zu ihrer Sprache passende Auswahl zu wählen. Der vom Übersetzer gewählte Standardeintrag wird verwendet, wenn deren Sprache benutzt wird, während Ihre eigene "Default"-Auswahl bei Englisch benutzt wird.

Verwenden Sie kein leeres Standardfeld. Wenn Sie keine Standardwerte verwenden möchten, verwenden Sie kein Feld "Default".

Falls Sie po-debconf benutzen (und das **sollten** Sie, lesen Sie *Seien Sie freundlich zu Übersetzern*), erwägen Sie, dieses Feld übersetzbar zu machen, wenn es nach Ihrer Meinung übersetzt werden könnte.

Ein Beispiel aus den Vorlagen des Pakets Geneweb:

```
Template: geneweb/lang
Type: select
__Choices: Afrikaans (af), Bulgarian (bg), Catalan (ca), Chinese (zh), Czech (cs),␣
↪Danish (da), Dutch (nl), English (en), Esperanto (eo), Estonian (et), Finnish (fi),␣
↪French (fr), German (de), Hebrew (he), Icelandic (is), Italian (it), Latvian (lv),␣
↪Norwegian (no), Polish (pl), Portuguese (pt), Romanian (ro), Russian (ru), Spanish␣
↪(es), Swedish (sv)
# This is the default choice. Translators may put their own language here
# instead of the default.
# WARNING : you MUST use the ENGLISH NAME of your language
# For instance, the French translator will need to put French (fr) here.
_Default: English[ translators, please see comment in PO files]
_Description: Geneweb default language:
```

Beachten Sie, dass die Benutzung von Klammern Kommentare in debconf-Feldern erlaubt. Beachten Sie außerdem, dass die Kommentare in den Dateien zu sehen sein werden, mit denen Übersetzer arbeiten.

Die Kommentare werden benötigt, da der _Default-Trick etwas verwirrend ist: Die Übersetzer können ihre eigene Wahl treffen.

6.7 Internationalisierung

Dieser Abschnitt enthält generelle Informationen, um Übersetzern die Arbeit zu erleichtern. Weitere Informationen für Übersetzer und Entwickler zum Thema Internationalisierung sind in der Dokumentation Internationalisierung und Lokalisierung verfügbar.

6.7.1 Handhabung von Debconf-Übersetzungen

Wie Portierer haben auch Übersetzer eine schwierige Aufgabe. Sie arbeiten an vielen Paketen und müssen mit vielen verschiedenen Paketbetreuern zusammenarbeiten, deren Muttersprache meist nicht Englisch ist. Sie sollten ihnen daher besondere Geduld entgegenbringen.

Das Ziel von `debconf` war die Vereinfachung der Paketkonfiguration für Betreuer und Anwender. Ursprünglich wurde die Übersetzung von Debconf-Vorlagen mit `debconf-mergetemplate` gehandhabt. Nun wird diese Technik jedoch missbilligt. Die Internationalisierung von `debconf` ist am besten mit dem Paket `po-debconf` zu erreichen. Diese Methode ist sowohl für Betreuer als auch für Übersetzer einfacher. Es werden Umwandlungsskripte bereitgestellt.

Wenn `po-debconf` benutzt wird, werden die Übersetzungen in `.po`-Dateien gespeichert (mit `gettext`-Übersetzungstechniken erstellt). Spezielle Vorlagendateien enthalten die Originalnachrichten und markieren, welche Felder übersetzbar sind. Wenn Sie den Wert eines übersetzbaren Feldes durch Aufruf von `debconf-updatepo` ändern, wird die Übersetzung für Übersetzer als aufmerksamkeitsbedürfend gekennzeichnet. Dann, während des Paketbaus, wird das Programm `dh_installdebconf` wie von Zauberhand dafür sorgen, dass alle Vorlagen zusammen mit den aktuellen Übersetzungen in die Binärpakete einfließen. Weitere Einzelheiten können Sie der Handbuchseite po-debconf(7) entnehmen.

6.7.2 Internationalisierte Dokumentation

Internationalisierte Dokumentation für Anwender ist wichtig, bereitet aber viel Mühe. Es gibt keine Möglichkeit, all diese Arbeit zu vermeiden, aber Sie können den Übersetzern einige Dinge erleichtern.

Falls Sie Dokumentationen in irgendwelchem Umfang betreuen, ist es für Übersetzer einfacher, wenn Sie Zugriff auf das Versionsverwaltungssystem haben. Dadurch können Übersetzer die Unterschiede zwischen zwei Versionen der Dokumentation anschauen, so dass sie beispielsweise sehen können, was neu übersetzt werden muss. Es wird empfohlen, dass die übersetzte Dokumentation eine Notiz darüber bereithält, auf welcher Revision des Quellcodes die Übersetzung basiert. Ein interessantes System wird von doc-check aus dem `debian-installer`-Paket bereitgestellt, das eine Übersicht über den Übersetzungsstatus für eine angegebene Sprache anzeigt. Dazu werden strukturierte Kommentare für die aktuelle Revision der zu übersetzenden Datei und für eine übersetzte Datei die Revision des Originals auf der die Übersetzung basiert, angezeigt. Möglicherweise möchten Sie dies anpassen und in Ihrem VCS-Bereich bereitstellen.

Falls Sie XML- oder SGML-Dokumentationen betreuen, wird dazu empfohlen, dass Sie jegliche sprachunabhängigen Informationen isolieren und diese als Entity in einer eigenen Datei definieren, die in allen verschiedenen Übersetzungen enthalten ist. Dies macht es beispielsweise viel einfacher, URLs über mehrere Dateien hinweg aktuell zu halten.

Einige Werkzeuge (z.B. `po4a`, `poxml` oder `translate-toolkit`) sind darauf spezialisiert, übersetzbares Material aus verschiedenen Formaten zu extrahieren. Sie erstellen PO-Dateien, ein für Übersetzer übliches Format, das eine Übersicht darüber gibt, was übersetzt werden muss, wenn das übersetzte Dokument aktualisiert wurde.

6.8 Optimale Vorgehensweisen für `debian/patches`

Debian Pakete können Fehler aus dem Originalcode enthalten, um die Sie sich kümmern müssen. In dem Quellformat "3.0 (quilt)" werden Patches unterhalb von `debian/patches/` gespeichert. Beim Auspacken des Quellpakets werden die Patches automatisch angewendet, so wie in `debian/patches/series` gelistet.

Patches sollten nach DEP-3 dokumentiert werden.

Es gibt einige Werkzeuge um das Verwalten von Patches zu automatisieren. Wenn Sie ein Quellpaket außerhalb eines Git-Repositories pflegen, dann ist wahrscheinlich `quilt` Ihre beste Option. Ansonsten sollten Sie überlegen sich auf die eingebauten Features von Git oder die Git-Paketierungs-Helfer zu verlassen, die Sie ggf. verwenden.

Insbesondere wenn Sie `git-buildpackage` für Pakete verwenden, sollten Sie die `gbp pq`-Kommandos verwenden um den Inhalt des Verzeichnisses `debian/patches/` zu verwalten.

A single patch can be created with e.g. `git format-patch -1 d33286c` from a single commit. Avoid using `git show` as it lacks the full headers.

Wenn die Fehlerkorrektur Upstream über mehrere Commits aufgeteilt ist, aber in Debian als einzelner Patch angewandt (und ausgeschlossen) werden soll, kann etwa der Befehl `git format-patch --stdout abc123..def456 > debian/patches/...` verwendet werden und das Feld `Bug` nur in der Commit-Message des ersten Commits im Patch hinzugefügt werden.

Wenn die Endung *.patch* zu der URL eines Commits oder Pull Requests bei GitHub oder eines Commits oder Merge Requests bei GitLab angehängt wird, verwendet die resultierende Patch-Datei das gleiche Format (als wäre sie mit `git format-patch` erstellt worden).

Denken Sie immer daran einen `Bug`-Header an die Patch-Beschreibung anzuhängen, so dass ein Leser dem Link folgen kann um zu sehen wo ein Fehler berichtet oder ein Patch eingereicht wurde. Wenn der Patch dafür da ist, um anhaltend von Upstream abzuweichen, hängen Sie den Header *Forwarded: not-needed* ans Ende der Beschreibung an.

6.9 Häufig vorkommende Paketierungssituationen

6.9.1 Pakete benutzen `autoconf/automake`

Die Dateien `config.sub` und `config.guess` von `autoconf` aktuell zu halten ist für Portierer kritisch, insbesondere auf eher unbeständigen Architekturen. Einige sehr gute Paketierungsvorgehensweisen für jegliche Pakete, die `autoconf` und/oder `automake` benutzen, wurden in `/usr/share/doc/autotools-dev/README.Debian.gz` aus dem Paket `autotools-dev` zusammengefasst. Es wird eindringlich geraten, diese README-Datei und die folgenden Empfehlungen zu lesen.

6.9.2 Bibliotheken

Das Paketieren von Bibliotheken gestaltet sich fast immer schwierig und dies aus unterschiedlichen Gründen. Die Richtlinien verhängen mehrere Beschränkungen, um ihre Pflege zu erleichtern und sicherzustellen, dass Upgrades so einfach wie möglich sind, wenn eine neue Originalversion herauskommt. Eine kaputte Bibliothek kann dazu führen, dass Dutzende davon abhängige Pakete beschädigt werden.

Gute Vorgehensweisen für das Paketieren von Bibliotheken wurden in der Anleitung zum Paketieren von Bibliotheken zusammengefasst.

6.9.3 Dokumentation

Achten Sie darauf, dass Sie den Richtlinien für Dokumentation folgen.

Falls Ihr Paket Dokumentation enthält, die aus XML oder SGML erstellt wurde, wird empfohlen in den Binärpaket(en) nicht die XML- oder SGML-Quellen mitzuliefern. Falls Anwender den Quellcode der Dokumentation möchten, sollten sie das Quellpaket herunterladen.

Die Richtlinie gibt an, dass die Dokumentation im HTML-Format weitergegeben werden sollte. Außerdem wird empfohlen, die Dokumentation im PDF-Format und als Klartext mitzuliefern, falls geeignet und falls die Ausgabe in einer vernünftigen Qualität möglich ist. Es ist allgemein jedoch nicht angemessen, Klartextversionen von Dokumentationen mitzuliefern, deren Quellformat HTML ist.

Bedeutende mitgelieferte Handbücher sollten sich selbst bei der Installation mit `doc-base` registrieren. Weitere Einzelheiten erhalten Sie in der Dokumentation des Pakets `doc-base`.

Die Debian-Richtlinien (Abschnitt 12.1) schreiben vor, dass Handbuchseiten jedem Programm, jedem Hilfswerkzeug und jeder Funktion beiliegen sollten und für andere Objekte, wie Konfigurationsdateien, wird dies nahegelegt. Falls die von Ihnen paketierte Arbeit nicht über eine solche Handbuchseite verfügt, dann überlegen Sie sich, eine zu schreiben, die Ihrem Paket beigefügt und an die Originalautoren gesandt wird.

Die Handbuchseiten müssen nicht direkt im Troff-Format geschrieben werden. Beliebte Quellformate sind Docbook, POD und reST, die mit `xsltproc`, `pod2man` beziehungsweise `rst2man` umgewandelt werden können. In

geringerem Maße kann außerdem das Programm `help2man` benutzt werden, um die Handbuchsseite rudimentär zu erstellen.

6.9.4 Besondere Pakettypen

Mehrere besondere Typen von Paketen haben spezielle Unter-Richtlinien und zugehörige Paketierungsregeln und Vorgehensweisen:

- Perl zugehörige Pakete haben eine Perl-Richtlinie. Einige Beispiele für Pakete, die dieser Richtlinie folgen, sind `libdbd-pg-perl` (binäres Perl-Modul) oder `libmldbm-perl` (architekturunabhängiges Perl-Modul).
- Python related packages have their Python policy; see `/usr/share/doc/python3/python-policy.txt.gz` in the `python3` package.
- Emacs zugehörige Pakete haben die Emacs-Richtlinie.
- Java zugehörige Pakete haben ihre Java-Richtlinie.
- OCaml zugehörige Pakete haben ihre eigene Richtlinie, die Sie unter `/usr/share/doc/ocaml/ocaml_packaging_policy.gz` im Paket `ocaml` finden können. Ein gutes Beispiel ist das Quellpaket `camlzip`.
- Pakete, die XML- oder SGML-DTDs bereitstellen, sollten konform zu den Empfehlungen im Paket `sgml-base-doc` sein.
- Rust-Paketierung wird im Debian Rust Team Book beschrieben.
- Pakete, die Services ("daemons") zur Verfügung stellen, sollten auf einem frisch installierten System funktionieren, insofern als das möglich ist ohne Sicherheit einzubüßen (z.B. sollte ein Webserver standardmäßig laufen und eine Beispielseite ausliefern, darf aber trotzdem keine unauthentifizierten sensitiven Operationen erlauben; Sie sollten überlegen, ob standardmäßig Seiten nur über das localhost Netzwerkinterface ausgeliefert werden).
- Web-Application-Pakete sollten darauf abzielen, ihre Abhängigkeiten (inklusive javascript) separat paketiert zu haben, und sollten notwendiges Setup für einen sicheren Basis-Betrieb ausführen (z.B. eine Datenbank anlegen, Konfigurationsdateien mit sicheren Standardeinstellungen ausliefern, Dateien an dem angebrachten Ort mit angebrachten Rechten installieren, etc). Als Beispiele können bestehende Pakete von Web-Applications herangezogen werden, z.B. dfsg-new-queue für Go, gitlab für ruby on rails, node-shiny-server für NPM. diaspora-installer ist ein Pseudopaket, welches diaspora herunterlädt (ebenso alle Laufzeit-Abhängigkeiten per rubygems) und es konfiguriert PostgreSQL und Nginx zu benutzen.

6.9.5 Architekturunabhängige Daten

Es ist unüblich größere Mengen an architekturunabhängiger Daten gemeinsam mit einem Programm zu paketieren. Als Beispiel hierzu seien Audiodateien, eine Symbolsammlung, Hintergrundmuster oder grafische Dateien genannt. Falls die Größe dieser Daten jedoch vernachlässigbar im Vergleich zum Rest des Pakets ist, ist es wahrscheinlich am besten, alles innerhalb eines einzelnen Paket zu paketieren.

Ist die Größe allerdings beachtlich, denken Sie darüber nach, sie in ein separates architekturunabhängiges Paket (`_all.deb`) auszulagern. Indem Sie dies tun, vermeiden Sie die im Prinzip unötige Vervielfältigung der gleichen Daten in zehn oder mehr .debs, eines pro Architektur! Obwohl dies einigen zusätzlichen Overhead zu den `Packages`-Dateien verursacht, spart es viel Plattenplatz auf den Debian-Spiegelservern. Das Heraus trennen architekturunabhängiger Daten vermindert auch die Ausführungszeit von `lintian` (siehe *Lint-Werkzeuge für Pakete*), wenn es für das ganze Debian-Archiv ausgeführt wird.

6.9.6 Eine bestimmte Locale wird während des Paketbaus benötigt

Falls Sie eine bestimmte Locale während des Baus benötigen, können Sie mittels dieses Tricks eine temporäre Datei erstellen:

Falls Sie `LOCPATH` auf die Entsprechung von `/usr/lib/locale` und `LC_ALL` auf den Namen der Locale setzen, die sie generieren, sollten Sie erreichen, was Sie möchten, ohne dass Sie Root sind. Etwas wie:

```
LOCALE_PATH=debian/tmpdir/usr/lib/locale
LOCALE_NAME=en_IN
LOCALE_CHARSET=UTF-8

mkdir -p $LOCALE_PATH
localedef -i $LOCALE_NAME.$LOCALE_CHARSET -f $LOCALE_CHARSET $LOCALE_PATH/$LOCALE_
↪NAME.$LOCALE_CHARSET

# Using the locale
LOCPATH=$LOCALE_PATH LC_ALL=$LOCALE_NAME.$LOCALE_CHARSET date
```

6.9.7 Machen Sie Übergangspakete deborphan-konform

Deborphan ist ein Programm, das Anwendern hilft, Pakete aufzuspüren, die sicher vom System entfernt werden können, d.h. diejenigen, von denen keine Pakete abhängen. Die Standardoperation ist, nur innerhalb der Abschnitte "libs" und "oldlibs" zu suchen, um Jagd auf unbenutzte Bibliotheken zu machen. Wenn aber das richtige Argument übergeben wird, versucht es auch, andere nutzlose Pakete zu erkennen.

Mit `--guess-dummy` versucht``deborphan`` beispielsweise, alle Übergangspakete zu durchsuchen, die für ein Upgrade benötigt wurden, aber jetzt entfernt werden können. Hierzu wird derzeit in der Kurzbeschreibung nach der Zeichenkette `dummy` oder `transitional` gesucht. Es ist jedoch besser, nach beiden Zeichenketten zu suchen, da es einige Dummy- oder Übergangspakete gibt, die anderen Zwecken dienen.

Wenn Sie also solch ein Paket erstellen, achten Sie bitte darauf den Text `transitional dummy package` in der Kurzbeschreibung zu verwenden. Um nach Beispielen zu suchen, führen Sie einfach `apt-cache search .|grep dummy` oder `apt-cache search .|grep transitional` aus.

Außerdem wird empfohlen, den Abschnitt in `oldlibs` und die Priorität in `optional` zu ändern, um die Arbeit von `deborphan` zu erleichtern.

6.9.8 Optimale Vorgehensweisen für `.orig.tar.{gz,bz2,xz}`-Dateien

Es gibt zwei Arten von Original-Quell-Tarballs: unberührten Quellcode und neu paketierten Quellcode der Originalautoren.

6.9.8.1 Unberührter Quellcode

Das charakteristische Merkmal eines unberührten Tarballs ist, dass die `.orig.tar.{gz,bz2,xz}`-Datei Byte für Byte identisch mit einem offiziell weitergegebenen Tarball des Originalautors ist.[1] Dies ermöglicht die Benutzung von Prüfsummen, um auf einfache Weise alle Änderungen zwischen Debians Version und der der Originalautoren zu prüfen, die in der Diff-Datei in Debian enthalten sind. Falls außerdem der Originalquellcode riesig ist, können Originalautoren und andere, die bereits den Original-Tarball haben, Download-Zeit sparen, falls sie Ihre Paketierung im Detail inspizieren möchten.

Es gibt keine allgemein anerkannten Leitlinien, denen Originalautoren betreffend der Verzeichnisstruktur innerhalb ihres Tarballs folgen, aber `dpkg-source` ist dennoch in der Lage, mit den meisten Tarballs von Originalautoren als unberührtem Quellcode umzugehen. Seine Strategie entspricht dem Folgenden:

1. Es entpackt den Tarball in eine leeres temporäres Verzeichnis mittels

   ```
   zcat path/to/packagename_upstream-version.orig.tar.gz | tar xf -
   ```

2. Falls das temporäre Verzeichnis danach nur ein Verzeichnis und keine anderen Dateien enthält, benennt `dpkg-source` dieses Verzeichnis in *Paketname_Originalversion*`(.orig)` um. Der Name des Verzeichnisses auf der obersten Ebene im Tarball ist ohne Bedeutung und geht verloren.

[1] Originalautoren können nicht daran gehindert werden, den von ihnen verteilten Tarball zu ändern, ohne die Versionsnummer zu erhöhen. Daher kann nicht gewährleistet werden, dass ein unberührter Tarball mit dem identisch ist, was die Originalautoren *aktuell* zu irgendeinem Zeitpunkt verteilen. Alles was erwartet werden kann, ist, dass es identisch mit etwas ist, das die Originalautoren einmal verteilt *haben*. Falls sich später ein Unterschied ergibt (etwa, wenn die Originalautoren merken, dass sie in ihrer Verteilung des Originals keine maximale Komprimierung nutzten und es dann erneut mit `gzip` packen), ist das einfach Pech. Da es keine brauchbare Möglichkeit gibt, ein neues `.orig.tar.{gz,bz2,xz}` für die gleiche Version hochzuladen, ist es sinnlos, dies als Fehler zu behandeln.

3. Andernfalls muss der Tarball der Originalautoren ohne ein sonst übliches Verzeichnis der obersten Ebene gepackt worden sein (Schande über den Originalautor!). In diesem Fall benennt `dpkg-source` das temporäre Verzeichnis *selbst* in *Paketname_Originalversion*`(.orig)` um.

6.9.8.2 Neu paketierter Originalquellcode

Sie **sollten** Pakete, wenn möglich, mit einem unberührten Quell-Tarball hochladen, aber es gibt viele Gründe, warum das manchmal nicht möglich ist. Dies ist der Fall, wenn die Originalautoren den Quellcode gar nicht als Gzip-gepackte Tar-Datei weitergeben oder falls der Tarball der Originalautoren nicht-DFSG-freies Material enthält, das Sie vor den Hochladen entfernen müssen.

In diesen Fällen muss der Entwickler selbst eine geeignete `.orig.tar.{gz,bz2,xz}`-Datei bauen. Solch einen Tarball nennen wir neu paketierten Originalquellcode. Beachten Sie, dass sich neu paketierter Originalquellcode von einem nativen Debian-Paket unterscheidet. Eine neu paketierte Quelle kommt mit Debian-spezifischen Änderungen in einem separaten `.diff.gz` oder `.debian.tar.{gz,bz2,xz}` daher und hat eine Versionsnummer, die sich aus der *Originalversion* und der *Debian-version* zusammensetzt.

Es könnte Gründe geben, aus denen es wünschenswert wäre, den Quellcode neu zu paketieren, obwohl die Originalautoren ein `.tar.{gz,bz2,xz}` verteilen, dass im Prinzip in seiner unberührten Form benutzt werden könnte. Der naheliegendste Grund ist, wenn *signifikante* Platzersparnis durch Neukomprimierung des Tar-Archivs oder Entfernen von wirklich nutzlosem Müll aus dem Qriginalarchiv erzielt werden kann. Handeln Sie hier nach eigenem Ermessen, aber seien Sie darauf vorbereitet, Ihre Entscheidung zu verteidigen, falls Sie einen Quellcode neu paketieren, der unberührt sein könnte.

Ein neu paketiertes `.orig.tar.{gz,bz2,xz}` Archiv

1. **sollte** im resultierenden Quellpaket dokumentiert werden. Detaillierte Informationen darüber, wie die neu gepackte Quelle erhalten wurde und wie diese reproduziert werden kann, sollten in `debian/copyright` bereitgestellt werden, idealerweise auf eine Weise, die automatisch mit uscan durchgeführt werden kann. Wenn das wirklich nicht funktioniert, stellen Sie zumindest ein `get-orig-source` Target in Ihrer `debian/rules` Datei bereit, das den Vorgang wiederholt, auch wenn dies in der Version 4.1.4 Debian Policy als veraltet markiert worden ist.

2. **sollte keine** Datei enthalten, die nicht von dem/den Originalautor(en) stammt oder deren Inhalt von Ihnen geändert wurde.[2]

3. **sollte** außer, wenn es aus rechtlichen Gründen unmöglich ist, die ganze Erstellungs- und Portierungsinfrastruktur aufbewahren, die vom Originalautor bereitgestellt wurde. Es ist zum Beispiel kein ausreichender Grund für das Weglassen einer Datei, wenn sie nur für die Erstellung unter MS-DOS benutzt wird. Gleichermaßen sollte ein `Makefile`, das vom Originalautor bereitgestellt wurde, nicht einmal dann weggelassen werden, wenn das erste, was Ihre `debian/rules` tut, das Überschreiben durch Ausführen eines Konfigurationsskripts ist.

   (*Begründung:* Es ist üblich für Debian-Anwender, die Software für nicht-Debian-Plattformen erstellen möchten, den Quellcode von einem Debian-Spiegel abzurufen, anstatt den Speicherort der ordnungsgemäßen Originaldistribution zu suchen).

4. **sollte** das Schema *Paketname-Upstream-Version*+`dfsg` (oder ein beliebiges Attributs-Suffix, das dem Tarball-Namen hinzugefügt wird) als Name für das Verzeichnis der obersten Ebene in seinem Tarball benutzen .

5. **should** be compressed with `xz` (or `gzip` or `bzip2`) with maximal compression.

6.9.8.3 Ändern binärer Dateien

Manchmal ist es nötig, binäre Dateien zu ändern, die im Original-Tarball enthalten sind oder binäre Dateien hinzuzufügen, die nicht darin enthalten sind. Dies wird vollständig unterstützt, wenn Sie Quellpakete im Format "3.0 (quilt)" benutzen. Lesen Sie die Handbuchseite dpkg-source(1), um weitere Einzelheiten zu erfahren. Wenn Sie das ältere Format "1.0" benutzen, können binäre Dateien nicht im `.diff.gz` gespeichert werden, daher müssen Sie

[2] Als besondere Ausnahme könnte, falls das Auslassen unfreier Dateien dazu führen würde, dass das Build des Quellcodes ohne Unterstützung aus dem Debian-Diff fehlschlägt, das Bearbeiten der Dateien anstelle des Weglassens unfreier Teile davon und/oder Erklären der Situation in einer `README.source`-Datei im Wurzelverzeichnis des Quelltextes angemessen sein. Drängen Sie in diesem Fall aber den Originalautor dazu, die unfreien Komponenten leichter von dem restlichen Quelltext separierbar zu machen.

eine mit uuencode (oder ähnlichem) kodierte Version der Datei(en) speichern und zur Erstellungszeit in debian/rules entschlüsseln (und an ihren offiziellen Platz verschieben).

6.9.9 Optimale Vorgehensweisen für Debug-Pakete

Ein Debug-Paket ist ein Paket, das zusätzliche Informationen enthält, die von gdb verwendet werden können. Da Debian diese Zusatzinformationen in Binärdateien standardmäßig entfernt, sind Debugging-Informationen wie Funktionsnamen und Zeilennummern dann nicht mehr verfügbar, wenn gdb auf Debian-Binärdateien ausgeführt wird. Mit Debug-Paketen können Benutzer, die diese zusätzlichen Debugging-Informationen benötigen, diese installieren, ohne ein reguläres System mit den in der Regel recht Speicherplatz intensiven Informationen aufzublähen.

Die Debug-Pakete enthalten separate Debugging-Symbole, die gdb beim Debuggen eines Programms oder einer Bibliothek im laufenden Betrieb finden und laden kann. Die Konvention in Debian besteht darin, diese Symbole in /usr/lib/debug /*path* abzulegen, wobei *path* der Pfad zur ausführbaren Datei oder Bibliothek ist. Zum Beispiel gehen Debugging-Symbole für /usr/bin/foo nach /usr/lib/debug/usr/bin/foo und Debugging-Symbole für /usr/lib/libfoo.so.1 gehen nach /usr/lib/debug/usr/lib/libfoo.so.1.

6.9.9.1 Automatisch erstellte Debug-Pakete

Debug-Symbolpakete können automatisch für jedes Binärpaket, das ausführbare Binärdateien enthält, generiert werden. Mit Ausnahme von Sonderfällen sollte es nicht mehr erforderlich sein, die alten manuell generierten zu verwenden. Der Paketname für ein automatisch generiertes Debug-Symbolpaket endet mit -dbgsym.

Die dbgsym-Pakete werden nicht in die regulären Archiven installiert, sondern in dedizierte Archive. Das heißt, wenn Sie die Debug-Symbole zum Debuggen benötigen, müssen Sie diese Archive zu Ihrer apt-Konfiguration hinzufügen und dann das entsprechende dbgsym-Paket installieren an dem Sie interessiert sind. Bitte lesen Sie https://wiki.debian.org/HowToGetABacktrace um mehr zu diesem Thema zu erfahren.

6.9.9.2 Manuelle -dbg Pakete

Vor dem Aufkommen der automatischen dbgsym-Pakete mussten Debug-Pakete manuell generiert werden. Der Name eines manuellen Debug-Pakets endet auf -dbg. Es wird empfohlen solche alten Legacy-Pakete nach Möglichkeit auf die neuen dbgsym-Pakete zu migrieren. Das Verfahren zum Konvertieren Ihres Pakets ist unter https://wiki.debian.org/AutomaticDebugPackages beschrieben. Im Wesentlichen wird jedoch der Schalter --dbgsym-migration = 'pkgname-dbg (<< currentversion~)' durch den Befehl dh_strip verwendet.

Manchmal ist es jedoch nicht möglich auf die neuen dbgsym-Pakete zu konvertieren, oder Sie werden auf die alten manuellen -dbg-Pakete in den Archiven stoßen, sodass Sie sich möglicherweise mit ihnen befassen müssen. Es wird nicht empfohlen, manuelle -dbg-Pakete für neue Pakete zu erstellen, es sei denn, die automatische Erstellung funktioniert aus irgendeinem Grund nicht.

Ein Grund könnte sein, dass Debug-Pakete einen ganzen speziellen Debugging-Build einer Bibliothek oder einer anderen Binärdatei enthalten. Normalerweise reicht es jedoch aus, Debugging-Informationen von den bereits erstellten Binärdateien zu trennen, dies spart zusätzlich Platz und Erstellungszeit.

Dies ist beispielsweise der Fall bei Debugging-Symbolen von Python-Erweiterungen. Derzeit ist der richtige Weg, um Python-Erweiterungs-Debug-Symbole zu verpacken, die Verwendung von -dbg-Paketen, wie unter https://wiki.debian.org/Python/DbgBuilds beschrieben.

Um -dbg-Pakete erstellen zu können müssen diese explizit in der Datei debian/control aufgeführt werden.

Die Debugging-Symbole können mit objcopy --only-keep-debug aus einer Objektdatei extrahiert werden. Dann kann die Objektdatei entfernt und objcopy --add-gnu-debuglink verwendet werden, um den Pfad zur Debugging-Symboldatei anzugeben. objcopy 1 erklärt ausführlich, wie das funktioniert.

Beachten Sie, dass Debug-Pakete von dem Paket abhängen sollten, für das Sie Debugging-Symbole bereitstellen und diese Abhängigkeit sollte mit einer Version versehen werden. Zum Beispiel:

```
Depends: libfoo (= ${binary:Version})
```

Der Befehl dh_strip in debhelper unterstützt das Erstellen von Debug-Paketen und kann die Benutzung von objcopy übernehmen, um die Debugging-Symbole für Sie zu filtern. Wenn Ihr Paket debhelper/9.20151219

oder neuer verwendet, müssen Sie nichts zusätzliches tun. `debhelper` generiert Debug-Symbol-Pakete (mit dem Schema Paket-`dbgsym`) für Sie ohne zusätzliche Änderungen an Ihrem Quellpaket.

6.9.10 Optimale Vorgehensweisen für Meta-Pakete

Ein Meta-Paket ist meist ein leeres Paket, das es vereinfacht, eine Zusammenstellung von Paketen zu installieren, die sich im Lauf der Zeit weiterentwickeln können. Dies wird erreicht, indem das Metapaket von allen Paketen der Zusammenstellung abhängt. Dank der Fähigkeiten von APT kann der Betreuer des Meta-Pakets die Abhängigkeiten anpassen und das System des Anwenders wird automatisch die zusätzlichen Pakete erhalten. Die weggelassenen Pakete, die automatisch installiert wurden, werden außerdem als Kandidaten für das Entfernen gekennzeichnet (und werden sogar durch `aptitude` automatisch entfernt). `gnome` und `linux-image-amd64` sind zwei Beispiele für Meta-Pakete (gebaut durch die Quellpakete `meta-gnome2` und `linux-latest`).

Die ausführliche Beschreibung des Meta-Pakets muss dessen Zweck klar dokumentieren, so dass die Benutzer wissen was sie verlieren, wenn sie das Paket entfernen. Es wird empfohlen, genau über die Auswirkungen zu informieren. Dies ist besonders für Meta-Pakete wichtig, die während der anfänglichen Installation installiert werden und nicht explizit durch den Benutzer installiert wurden. Diese sind oft wichtig für das reibungslose Upgrade des Systems und der Benutzer sollte davon abgehalten werden, diese zu entfernen, um mögliche Schäden zu vermeiden.

KAPITEL 7

Jenseits von Paketierung

Debian ist weit mehr als nur Software zu paketieren und diese Pakete zu verwalten. Dieses Kapitel enthält Informationen über Wege, oft wirklich kritische Wege, fernab von einfachem Erstellen und Verwalten von Paketen zu Debian beizutragen.

Als eine Organisation von Freiwilligen verlässt sich Debian auf das Ermessen der Auswahl seiner Mitglieder, woran sie arbeiten möchten und was aus deren Sicht die kritischste Sache ist, in die sie Zeit investieren.

7.1 Fehler berichten

Bitte reichen Sie Fehlerberichte ein, wenn Sie Fehler in Debian-Paketen finden. Tatsächlich bilden Debian-Entwickler oftmals die erste Reihe der Tester. Fehler in den Paketen anderer Entwickler zu finden und zu melden, erhöht die Qualität von Debian.

Lesen Sie die Anweisungen zum Melden von Fehlern in der Debian-Fehlerdatenbank.

Versuchen Sie, den Fehlerbericht von einem normalen Benutzerkonto zu senden, auf dem Sie wahrscheinlich E-Mails empfangen können, so dass Leute, die weitere Informationen über den Fehler benötigen, Sie erreichen können. Senden Sie keine Fehlerberichte als Root.

Sie können ein Werkzeug wie reportbug 1 benutzen, um Fehlerberichte zu senden. Es kann den Prozess automatisieren und generell erleichtern.

Prüfen Sie, dass der Fehlerbericht nicht bereits schon gegen das Paket eingereicht wurde. Jedes Paket hat eine Fehlerliste, die einfach unter `https://bugs.debian.org/`*Paketname* einsehbar ist. Hilfswerkzeuge wie querybts 1 können Sie ebenfalls mit diesen Informationen versorgen (und `reportbug` wird normalerweise auch vor dem Versenden `querybts` aufrufen).

Versuchen Sie, Ihre Fehlerberichte an die ordnungsgemäße Stelle zu lenken. Wenn Ihr Fehlerbericht zum Beispiel von einem Paket handelt, das Dateien eines anderen Pakets überschreibt, prüfen Sie die Fehlerliste *beider* Pakete, um das Einreichen doppelter Fehlerberichte zu vermeiden.

Um zusätzliche Anerkennung zu erhalten, vereinigen Sie Fehler, die mehr als einmal berichtet wurden oder kennzeichnen Sie Fehler als "fixed", die bereits behoben wurden. Beachten Sie, dass Sie den Fehlerbericht nicht tatsächlich schließen sollten wenn Sie weder der Absender des Fehlers noch der Betreuer des Pakets sind (es sei denn, Sie versichern sich der Erlaubnis des Betreuers).

Von Zeit zu Zeit möchten Sie vielleicht prüfen, was aus den Fehlerberichten geworden ist, die Sie versandt haben. Nutzen Sie diese Gelegenheit, um diejenigen zu schließen, die Sie nicht mehr reproduzieren können. Um herauszufinden, welche Fehlerberichte Sie versandt haben, besuchen Sie `https://bugs.debian.org/from:`*Ihre-E-Mail-Adresse*.

7.1.1 Viele Fehler auf einmal berichten (Masseneinreichung von Fehlern)

Das Berichten einer großen Anzahl von Fehlern für dasselbe Problem für eine große Zahl von Paketen — d.h. mehr als zehn — ist eine nicht erwünschte Vorgehensweise. Unternehmen Sie alle nötigen Schritte, um das massenhafte Versenden von Fehlerberichten tunlichst zu vermeiden. Prüfen Sie zum Beispiel, ob das Problem automatisiert werden kann, indem Sie eine neue Prüfung zu `lintian` hinzufügen, so dass eine Fehlermeldung oder Warnung ausgegeben wird.

Falls Sie mehr als zehn Fehler auf einmal zum gleichen Thema berichten, wird empfohlen, dass Sie eine E-Mail an `debian-devel@lists.debian.org` senden, in der Sie Ihre Absichten darlegen, bevor Sie den Bericht versenden und die Tatsache im Betreff Ihrer Mail erwähnen. Dies wird anderen Entwicklern ermöglichen zu prüfen, ob der Fehler ein echtes Problem darstellt. Zusätzlich wird es helfen, eine Situation zu vermeiden, in der mehrere Betreuer simultan beginnen, den gleichen Fehlerbericht einzureichen.

Bitte benutzen Sie das Programm `dd-list` und falls geeignet `whodepends` (aus dem Paket `devscripts`), um eine Liste aller betroffenen Pakete zu generieren und fügen sie die Ausgabe in Ihre Mail an `debian-devel@lists.debian.org` ein.

Beachten Sie, wenn Sie viele Fehlerberichte mit dem gleichen Betreff senden möchten, dass Sie den Fehlerbericht an `maintonly@bugs.debian.org` senden sollten, so dass der Fehlerbericht nicht an die Fehlerverteilungs-Maillingliste weitergeleitet wird.

Das Programm `mass-bug` (aus dem Paket `devscripts`) kann benutzt werden um Bugreports gegen eine Liste von Paketen zu öffnen.

7.1.1.1 Usertags

Vielleicht möchten Sie BTS-Usertags (benutzerdefinierte Kennzeichnungen) benutzen, wenn Sie Fehlerberichte an eine größere Anzahl Pakete senden. Diese Kennzeichnungen sind den normalen Kennzeichen wie "patch" oder "wishlist" ähnlich, unterscheiden sich aber darin, dass sie benutzerdefiniert sind und einen Namensraum belegen, der einzigartig für einen bestimmten Benutzer ist. Dies ermöglicht mehreren Gruppierungen von Entwicklern, den gleichen Fehler benutzerdefiniert auf unterschiedliche Arten zu kennzeichnen, ohne Konflikte zu verursachen.

Um beim Einreichen von Fehlerberichten Usertags hinzuzufügen, geben Sie die Pseudokopfzeilen `User` und `Usertags` an:

```
To: submit@bugs.debian.org
Subject: title-of-bug

Package: pkgname
[ ... ]
User: email-addr
Usertags: tag-name [ tag-name ... ]

description-of-bug ...
```

Beachten Sie, dass Kennzeichnungen durch Leerzeichen getrennt werden und keine Unterstriche enthalten dürfen. Falls Sie Fehlerberichte für eine spezielle Gruppe oder ein Team einreichen, wird empfohlen, dass Sie bei `User` eine passende Mailingliste eintragen, nachdem Sie Ihre Absicht dort geschildert haben.

Um Fehler mit einem bestimmten Usertag anzusehen, benutzen Sie `https://bugs.debian.org/cgi-bin/pkgreport.cgi?users=`*E-Mail-Adresse*`&tag=`*Kennzeichen*.

7.2 Qualitätssicherungsbestreben

7.2.1 Tägliche Arbeit

Auch wenn es eine eigene Gruppe für Qualitätssicherung gibt, sind QS-Aufgaben nicht ausschließlich dieser Gruppe vorbehalten. Sie können sich an dieser Aufgabe beteiligen, indem Sie Ihre Pakete so fehlerfrei und lintian-rein (siehe *lintian*) wie möglich halten. Falls Sie finden, dies sei unmöglich, dann sollten Sie darüber nachdenken, einige Ihrer Pakete zu verwaisen (siehe *Verwaisen von Paketen*). Alternativ könnten Sie andere Leute um Hilfe bitten, um den Rückstand, den Sie bei den Fehlern haben, aufzuholen (Sie können auf `debian-qa@lists.debian.org`

oder debian-devel@lists.debian.org nach Hilfe fragen). Gleichzeitig können Sie sich nach Mitbetreuern umsehen (siehe *Gemeinschaftliche Verwaltung*).

7.2.2 Treffen zur gemeinschaftlichen Behebung von Fehlern (Bug-Squashing-Partys)

Von Zeit zu Zeit organisiert die QS-Gruppe Bug-Squashing-Partys, um so viele Probleme wie möglich zu beseitigen. Sie werden auf debian-devel-announce@lists.debian.org angekündigt und in der Ankündigung wird erklärt, auf welchem Bereich der Fokus der Party liegt: Üblicherweise konzentriert man sich auf Release-kritische Fehler, aber es kann auch vorkommen, dass entschieden wird, bei der Fertigstellung eines Haupt-Upgrades zu helfen (wie einer neuen perl-Version, die ein Neukompilieren aller binären Module erfordert).

The rules for non-maintainer uploads differ during the parties because the announcement of the party is considered prior notice for NMU. If you have packages that may be affected by the party (because they have release critical bugs for example), you should send an update to each of the corresponding bugs to explain their current status and what you expect from the party. If you don't want an NMU, or if you're only interested in a patch, or if you will deal with the bug yourself, please explain that in the BTS.

Teilnehmer der Party haben besondere Regeln für NMUs. Sie können NMUs ohne vorherige Ankündigung durchführen, falls sie mindestens nach DELAYED/3-day hochladen. Alle anderen NMU-Regeln gelten wie üblich; sie sollten den Patch des NMUs an das BTS senden (an einen der offenen Fehler, der durch den NMU behoben wird oder an einen neuen Fehler, der als "fixed" gekennzeichnet wird). Sie sollten außerdem die besonderen Wünsche des Betreuers respektieren.

Falls Sie sich nicht sicher fühlen, einen NMU durchzuführen, senden Sie nur einen Patch an das BTS. Das ist weitaus besser als ein beschädigter NMU.

7.3 Andere Paketbetreuer kontaktieren

Während Ihres Lebens innerhalb von Debian werden Sie aus verschiedenen Gründen Kontakt zu anderen Betreuern haben. Sie möchten vielleicht neue Wege der Kooperation zwischen einer Zusammenstellung verwandter Pakete diskutieren oder einfach jemanden daran erinnern, dass eine neue Originalversion verfügbar ist, die Sie benötigen.

Die E-Mail-Adresse eines Paketbetreuers heraussuchen zu müssen, kann störend sein. Glücklicherweise gibt es einen einfachen E-Mail-Alias, *Paket*@packages.debian.org, der eine Möglichkeit bietet, dem Betreuer eine Mail zu schicken, ganz gleich wie seine E-Mail-Adresse (oder Adressen) auch sein mag. Ersetzen Sie *Paket* durch den Namen eines Quell- oder Binärpakets.

Möglicherweise sind Sie außerdem daran interessiert, Personen zu kontaktieren, die ein bestimmtes Quellpaket mittels *Das Debian-Paketverfolgungssystem* abonniert haben. Sie können dazu die E-Mail-Adresse *Paket*@packages.qa.debian.org benutzen.

7.4 Sich mit inaktiven und/oder nicht erreichbaren Paketbetreuern beschäftigen

Falls Sie bemerken, dass es einem Paket an Pflege mangelt, sollten Sie prüfen, dass der Betreuer aktiv ist und weiter an seinen Paketen arbeitet. Es ist möglich, dass er nicht mehr aktiv ist, sich aber nicht aus dem System abgemeldet hat. Andererseits ist es auch möglich, dass er nur eine Erinnerung braucht.

There is a simple system (the MIA database) in which information about maintainers who are deemed Missing In Action is recorded. When a member of the QA group contacts an inactive maintainer or finds more information about one, this is recorded in the MIA database. This system is available in the directory /org/qa.debian.org/mia on the host qa.debian.org. The directory contains the tool mia-query with which the database can be queried. Use /org/qa.debian.org/mia/mia-query --help to see how to query the database. For example, to find out information about Debian Developer named Maria you could run:

```
ssh qa.debian.org /org/qa.debian.org/mia/mia-query --name Maria
```

If you find that no information has been recorded about an inactive maintainer yet, or that you can add more information, you should generally proceed as follows.

Im ersten Schritt kontaktieren Sie den Betreuer höflich und warten eine angemessene Zeit auf eine Antwort. Es ist ziemlich schwer zu definieren, was eine angemessene Zeit ist, aber es ist wichtig zu berücksichtigen, dass das wahre Leben manchmal sehr hektisch ist. Eine Möglichkeit damit umzugehen wäre es, nach zwei Wochen eine Erinnerung zu senden.

Eine nicht funktionierende E-Mail-Adresse ist eine Verletzung der Debian Richtlinien. Falls eine E-Mail nicht zustellbar ist, melden Sie dies bitte als einen Fehler gegen das Paket, und informieren Sie das MIA-Team.

Falls der Betreuer nicht innerhalb von vier Wochen (einem Monat) antwortet, kann davon ausgegangen werden, dass wahrscheinlich keine Antwort mehr kommt. Falls dies geschieht, sollten Sie weiter nachforschen und versuchen so viele nützliche Informationen wie möglich über den betreffenden Betreuer zu sammeln. Dies beinhaltet:

- Die `echelon`-Informationen, die über debian.org Developers LDAP Search verfügbar sind. Sie geben an, wann der Entwickler zum letzten Mal an eine Debian-Mailingliste geschrieben hat. (Dies umfasst auch Mails über Uploads, die über die Liste `debian-devel-changes@lists.debian.org` verteilt wurden.) Denken Sie außerdem daran zu prüfen, ob der Betreuer in der Datenbank als im Urlaub befindlich markiert ist.
- Die Anzahll der Pakete, für die dieser Betreuer verantwortlich ist und den Zustand dieser Pakete. Hauptsächlich, ob es Release-kritische Fehler gibt, die seit Jahren offen sind. Ferner wie viele Fehler es im Allgemeinen sind. Ein weiterer wichtiger Teil der Informationen ist, ob für die Pakete NMUs durchgeführt werden und wenn von wem.
- Gibt es irgendeine Aktivität des Betreuers außerhalb von Debian? Er könnte zum Beispiel aktuell etwas an eine Nicht-Debian-Mailingliste oder an Newsgroups geschrieben haben.

Ein ziemliches Problem sind Pakete, die gesponsert wurden — der Betreuer ist kein offizieller Debian-Entwickler. Die `echelon`-Informationen sind nicht für gesponserte Leute verfügbar, so dass Sie beispielsweise den Debian-Entwickler finden und kontaktieren müssen, der das Paket tatsächlich hochgeladen hat. Angenommen, das Paket wurde signiert, dann ist er dennoch für das Paket verantwortlich und weiß wahrscheinlich, wie es mit der Person aussieht, für die er das Paket sponserte.

Es ist außerdem erlaubt, eine Anfrage an `debian-devel@lists.debian.org` zu senden und zu fragen, ob jemand etwas über den Verbleib des vermissten Betreuers weiß. Bitte senden Sie der Person, um die es geht, per Cc: eine Kopie.

Sobald Sie all dies gesammelt haben, können Sie `mia@qa.debian.org` kontaktieren. Die Leute hinter diesem Alias werden die von Ihnen bereitgestellten Informationen nutzen, um zu entscheiden, wie verfahren wird. Beispielsweise könnten sie eines oder alle Pakete des Betreuers verwaisen. Falls für ein Paket ein NMU durchgeführt wurde, könnten sie es vorziehen, vor dem Verwaisen des Pakets denjenigen zu kontaktieren, von dem der NMU durchgeführt wurde — vielleicht ist diese Person daran interessiert, das Paket zu übernehmen.

Eine abschließende Bemerkung: Bitte denken Sie daran immer höflich zu bleiben. Alle hier sind Freiwillige und können nicht sämtliche Zeit Debian widmen. Außerdem wissen Sie nichts über die Situation der beteiligten Person. Vielleicht ist sie ernsthaft erkrankt oder sogar verstorben — Sie wissen nicht, wer auf der Empfängerseite ist. Stellen Sie sich vor, wie sich ein Verwandter fühlt, wenn er die E-Mail des Verstorbenen liest und eine sehr unhöfliche, böse oder vorwurfsvolle Nachricht findet!

Andererseits, trotz dass wir alle Freiwillige sind, geht ein Paketbetreuer auch eine Verpflichtung ein und hat deshalb auch eine Verantwortung, das Paket zu pflegen. So können Sie die Wichtigkeit eines übergeordneten Wohls betonen — falls die Betreuer keine Zeit oder kein Interesse mehr haben, sollten sie loslassen und das Paket jemandem mit mehr Zeit übergeben.

Wenn Sie Interesse daran haben im MIA Team mitzuarbeiten dann schauen Sie bitte in die Datei *README*` in `/org/qa.debian.org/mia` auf `qa.debian.org`. In dieser Datei sind die technischen Details als auch der MIA Prozess dokumentiert. Zur Kontaktaufnahme mit dem MIA Team schreiben Sie bitte an E-Mail an `mia@qa.debian.org`.

7.5 Umgang mit wiederholt nachlässigen Debian-Entwicklern

Von Debian-Entwicklern wird erwartet sich an eine Reihe von technischen und nicht-technischen Regeln zu halten (zum Beispiel die Benutzungsrichtlinien für Debian-Rechner und der Debian Code of Conduct).

To err is human and one should generally assume good intent, but repeated technical mistakes and/or disrespectful interactions, combined with a dismissive attitude towards Debian policies should be referred to the Debian Community Team. Most escalations to the Community Team are resolved amicably, but extreme cases may escalate further to the Debian Project Leader (leader@debian.org) and the Debian Account Managers (da-manager@debian.org) and could ultimately lead to expulsion from the Debian project.

7.6 Zusammenwirken mit zukünftigen Debian-Entwicklern

Debians Erfolg hängt von der Fähigkeit ab, neue und talentierte Entwickler für das Projekt zu gewinnen und auch zu halten. Wenn Sie ein erfahrener Entwickler sind, wird Ihnen empfohlen, sich am Prozess neue Entwickler heranzuziehen zu beteiligen.

7.6.1 Pakete sponsern

Sponsern eines Pakets bedeutet, ein Paket für einen Betreuer hochzuladen, der nicht in der Lage ist, dies selbst zu tun. Es ist keine belanglose Angelegenheit, der Sponsor muss die Paketierung überprüfen und dafür sorgen, dass sie der hohen Qualität entspricht, die Debian anstrebt.

Debian-Entwickler können Pakete sponsoren. Debian-Betreuer können dies nicht.

Der Prozess, ein Paket zu sponsern ist:

1. The maintainer prepares a source package (`.dsc`) and puts it online somewhere (like on mentors.debian.net) or even better, provides a link to a public VCS repository (see *salsa.debian.org: Git Repositorys und kollaborative Entwicklungsplattform*) where the package is maintained.
2. Der Sponsor lädt das Quellpaket herunter (oder führt einen Checkout durch).
3. Der Sponsor prüft das Quellpaket. Falls er Probleme entdeckt, informiert er den Betreuer und bittet ihn, eine korrigierte Version bereitzustellen (der Prozess beginnt von vorn mit dem ersten Schritt).
4. Der Sponsor konnte kein verbliebenes Problem finden. Er erstellt das Paket, signiert es und lädt es nach Debian hoch.

Bevor Sie die Einzelheiten erforschen, wie ein Paket gesponsert wird, sollten Sie sich selbst fragen, ob das vorgeschlagene Paket einen Mehrwert für Debian und dessen Nutzer bringt.

Es gibt keine einfache Regel, um diese Frage zu beantworten, sie kann von vielen Faktoren abhängen: Ist die Code-Basis der Originalautoren ausgereift und nicht voller Sicherheitslücken? Gibt es bereits existierende Pakete, die die gleiche Aufgabe erledigen können und wie sind diese im Vergleich zu diesem neuen Paket? Gab es für dieses neue Paket eine Nachfrage durch Anwender und wie groß ist die Benutzerbasis? Wie aktiv sind die Entwickler des Originalsoftware?

Sie sollten außerdem sicherstellen, dass der zukünftige Betreuer ein guter Betreuer sein wird. Hat er bereits etwas Erfahrung mit anderen Paketen? Falls ja, leistet er bei ihnen gute Arbeit (überprüfen Sie einige Fehlerberichte)? Ist er vertraut mit dem Paket und dessen Programmiersprache? Verfügt er über die für dieses Paket nötigen Fähigkeiten? Falls nicht: ist er in der Lage, sie zu erlernen?

Es ist auch eine gute Idee zu wissen, wo der neue Betreuer im Bezug zu Debian steht: Stimmt er Debians Philosophie zu und beabsichtigt sich der Debian Gemeinschaft anzuschließen? Angesichts der Tatsache, wie einfach es ist, Debian-Mitglied zu werden, möchten Sie möglicherweise nur Personen sponsern, die eine Mitgliedschaft planen. Auf diese Weise wissen Sie von Anfang an, dass Sie nicht auf unbestimmte Zeit als Sponsor auftreten müssen.

7.6.1.1 Ein neues Paket sponsern

Neue Betreuer haben gewöhnlich bestimmte Schwierigkeiten bei der Erstellung von Debian-Paketen — dies ist ziemlich verständlich. Sie werden nicht immer alles richtig machen. Das ist der Grund, weshalb das Sponsern eines brandneuen Pakets in Debian eine sorgfältige Überprüfung notwendig macht. Manchmal sind mehrere Wiederholungen nötig, bis das Paket gut genug ist, um ins Debian Archive geladen werden zu können. Daher impliziert ein Sponsor zu sein ebenfalls auch ein Mentor zu sein.

Sponsern Sie ein Paket nie, ohne es zu überprüfen. Die Überprüfung eines neuen Pakets, die von den Ftpmasters vorgenommen wird, stellt nur sicher, dass die Software wirklich frei ist. Natürlich kommt es vor, dass sie über

Paketierungsprobleme stolpern, aber das sollte wirklich nicht passieren. Es ist Ihre Aufgabe, dafür zu sorgen, dass das hochgeladene Paket mit Debians Richtlinien für freie Software übereinstimmt und von guter Qualität ist.

Das Erstellen des Pakets und Prüfen der Software ist Teil der Überprüfung, aber es reicht nicht aus. Der Rest dieses Abschnitts enthält eine unvollständige Liste von Punkten, die Sie bei Ihrer Überprüfung testen sollten.[1]

- Prüfen Sie, ob der bereitgestellte Original-Tarball derselbe ist wie der, den der Originalautor verteilt (wenn die Quellen neu für Debian gepackt wurden, erstellen Sie selbst den geänderten Tarball).
- Führen Sie `lintian` aus (siehe *lintian*). Sie werden so gängige Probleme finden. Verifizieren Sie, dass jegliche `lintian`-overrides des Betreuers vollständig gerechtfertigt sind.
- Run `licensecheck` (see *licensecheck*) and verify that `debian/copyright` seems correct and complete. Look for license problems (like files with "All rights reserved" headers, or with a non-DFSG compliant license). `grep -ri` is your friend for this task.
- Erstellen Sie das Paket mit `pbuilder` (oder einem ähnlichen Werkzeug, siehe *pbuilder*), um sicherzustellen, dass die Build-Abhängigkeiten vollständig sind.
- Lesen Sie `debian/control` Korrektur: Folgt es den optimalen Vorgehensweisen (siehe *Optimale Vorgehensweisen für debian/control*)? Sind die Abhängigkeiten vollständig?
- Lesen Sie `debian/rules` Korrektur: Folgt es den optimalen Vorgehensweisen (siehe *Optimale Vorgehensweisen für debian/rules*)? Sehen Sie mögliche Verbesserungen?
- Lesen Sie die Betreuerskripte (`preinst`, `postinst`, `prerm`, `postrm`, `config`) Korrektur: Werden `preinst`/`postrm` funktionieren, wenn die Abhängigkeiten nicht installiert sind? Sind alle Skripte idempotent (d.h. können sie ohne Konsequenzen mehrmals ausführt werden)?
- Überprüfen Sie jede Änderung an Originaldateien (entweder in `.diff.gz`, in `debian/patches/` oder direkt in den Tarball `debian` für Binärdateien eingebettet). Sind sie gerechtfertigt? Sind sie ordentlich dokumentiert (mit DEP-3 für Patches)?
- Fragen Sie sich für jede Datei, warum diese Datei dort ist und ob dies der richtige Weg ist, das gewünschte Ergebnis zu erzielen. Folgt der Betreuer den optimalen Vorgehensweisen beim Paketieren (siehe *Optimale Vorgehensweise beim Paketieren*)?
- Erstellen Sie die Pakete, installieren Sie sie und probieren Sie die Software aus. Stellen Sie sicher, dass Sie die Pakete entfernen und vollständig entfernen können. Testen Sie sie eventuell mit `piuparts`.

Falls die Kontrolle keine Probleme offenbarte, können Sie das Paket erstellen und nach Debian hochladen. Denken Sie daran, dass, obwohl Sie nicht der Betreuer sind, Sie als Sponsor trotzdem auch für das verantwortlich sind, was Sie nach Debian hochladen. Daher sollten Sie das Paket durch das *Das Debian-Paketverfolgungssystem* im Auge zu behalten.

Beachten Sie, dass Sie das Quellpaket nicht verändern sollten, um Ihren Namen in die Datei `changelog` oder `control` einzutragen. Das Feld `Maintainer` der Dateien `control` und `changelog` sollte die Person aufführen, die das Paket erstellte, d.h. den Gesponserten. Auf diese Art wird er die gesamten Nachrichten vom BTS erhalten.

Stattdessen sollten Sie `dpkg-buildpackage` anweisen, Ihren Schlüssel für die Signatur zu benutzen. Sie erreichen dies mit der Option `-k`:

```
dpkg-buildpackage -kKEY-ID
```

Falls Sie `debuild` und `debsign` benutzen, können sie das sogar permanent in `~/.devscripts` konfigurieren:

```
DEBSIGN_KEYID=KEY-ID
```

7.6.1.2 Eine Aktualisierung eines existierenden Pakets sponsern

Sie werden normalerweise davon ausgehen, dass das Paket bereits eine vollständige Überprüfung durchlief. Daher werden Sie, anstatt dies nochmals erneut zu tun, nur die Unterschiede zwischen der aktuellen und der neu vom

[1] Sie können weitere Überprüfungen im Wiki finden, wo verschiedene Entwickler ihre eigenen Sponsoring-Prüflisten miteinander teilen.

Betreuer vorbereiteten Version sorgsam analysieren. Falls Sie die anfängliche Überprüfung nicht selbst durchgeführt haben, könnten Sie auch noch einen genaueren Blick darauf werfen, nur für den Fall, dass der erste Prüfer nachlässig war.

Damit Sie die Unterschiede untersuchen können, benötigen Sie beide Versionen. Laden Sie die aktuelle Version des Quellpakets herunter (mit `apt-get source`) und erstellen Sie es (oder laden Sie die aktuellen Binärpakete mit `aptitude download` herunter). Laden Sie das Quellpaket zum Sponsern (üblicherweise mit `dget`).

Lesen Sie das Änderungsprotokoll, damit Sie wissen, was Sie während der Überprüfung erwartet. Das Hauptwerkzeug, das Sie benutzen werden, ist `debdiff` (bereitgestellt vom Paket `devscripts`). Sie können es mit zwei Quellpaketen (`.dsc`-Dateien), zwei Binärpaketen oder zwei `.changes`-Dateien (dann wird es alle Binärpakete vergleichen, die in `.changes` aufgeführt sind) ausführen.

Falls Sie die Quellpakete vergleichen (ausschließlich der Originaldateien im Fall einer neuen Originalversion, zum Beispiel durch Filtern der Ausgabe von `debdiff` mit `filterdiff -i '*/debian/*'`), müssen Sie alle Änderungen verstehen und sie sollten ordentlich im Debian-Änderungsprotokoll dokumentiert sein.

Falls alles in Ordnung ist, erstellen Sie das Paket und vergleichen Sie die Binärpakete, um zu prüfen, ob die Änderungen am Quellpaket keine unerwarteten Folgen haben (wie einige versehentlich entfallenen Dateien, fehlende Abhängigkeiten etc.)

Sie möchten möglicherweise das Package-Tracking-System nutzen (siehe *Das Debian-Paketverfolgungssystem*), um zu überprüfen, ob der Betreuer nichts wichtiges versäumt hat. Möglicherweise gibt es Übersetzungsaktualisierungen, die im BTS liegen und hätten integriert werden können. Möglicherweise wurde ein NMU des Pakets durchgeführt und der Betreuer vergaß, die Änderungen des NMUs in sein Paket einzubauen. Vielleicht ist ein Release-kritischer Fehler unbehandelt geblieben und dies blockiert die Migration nach `Testing`. Falls Sie etwas finden, was (besser) erledigt werden könnte, ist es an der Zeit, ihm dies mitzuteilen, so dass er es dies das nächste Mal besser machen kann und dass er seine Verantwortlichkeiten besser versteht.

Falls Sie kein bedeutendes Problem gefunden haben, laden Sie die neue Version hoch. Andernfalls bitten Sie den Betreuer, Ihnen eine korrigierte Version zur Verfügung zu stellen.

7.6.2 Einrichten von Uploadberechtigungen für DM

Ein Debian Developer kann Uploadberechtigungen an einem spezifischen Paket für einen Debian Maintainer einrichten sobald der Schlüssel des Debian Maintainers in den Debian Maintainer Schlüsselring aufgenommen worden ist. Dazu muss der Developer ein signiertes DAK-Kommando an ftp.upload.debian.org schicken wie in der FTP-Master Ankündigung an Debian-Devel beschrieben worden ist.

Dieser Prozess kann durch die Hilfe vom `dcut` Kommando aus dem Paket `dput-ng` vereinfacht werden. Hinweis, das funktioniert nicht mit dem `dcut` Kommando aus dem Paket `dput`!

Zum Beispiel:

```
dcut dm --uid 0xfedcba9876543210 --allow nano --deny bash
```

Wenn sich der Schlüssel des DM noch nicht im Paket vom Schlüsselring, aber im lokalen Schlüsselbund des DD befindet, verwenden Sie die Option „--force“ und den Fingerabdruck, ohne Leerzeichen und in diesem speziellen Fall ohne das Präfix 0x und ausschließlich in Großbuchstaben:

```
dcut --force dm --uid FEDCBA9876543210FEDCBA9876543210 --allow nano
```

7.6.3 Neue Entwickler befürworten

Lesen Sie die Seite Einen zukünftigen Entwickler befürworten auf der Debian-Website.

7.6.4 Handhabung von Bewerbungen neuer Betreuer

Lesen Sie bitte die Checkliste für Bewerbungsleiter auf der Debian-Website.

KAPITEL 8

Internationalisierung und Übersetzungen

Debian unterstützt eine immer größer werdende Zahl natürlicher Sprachen. Selbst wenn Englisch Ihre Muttersprache ist und Sie keine andere Sprache sprechen, gehört es zu Ihren Pflichten als Paketbetreuer, die Probleme der Internationalisierung zu kennen (abgekürzt I18n, weil 18 Buchstaben zwischen "i" und "n" im englischen Wort "internationalization" stehen). Daher sollten Sie sogar, wenn Sie mit rein englischen Programmen klarkommen, das meiste in diesem Kapitel lesen.

i18n means modification of software or related technologies so that software can potentially handle multiple languages, customs, and other differences, while L10N (localization) means implementation of a specific language for already-internationalized software.

L10n und I18n sind miteinander verbunden, aber die Schwierigkeiten bezogen auf jeweils eines davon sind sehr unterschiedlich. Es ist nicht wirklich schwer, einem Programm das Wechseln der Sprache zu erlauben, in dem Texte basierend auf den Benutzereinstellungen angezeigt werden, aber es verschlingt viel Zeit, diese Nachrichten tatsächlich zu übersetzen. Andererseits ist es trivial, die Zeichenkodierung einzustellen, aber den Code so anzupassen, dass mehrere Zeichenkodierungen benutzt werden können, ist ein wirklich großes Problem.

Abgesehen von den I18n-Problemen, für die keine allgemeine Anleitung gegeben werden kann, gibt es tatsächlich keine Infrastruktur für L10n innerhalb von Debian, die mit dem Buildd-Mechanismus für die Portierung vergleichbar ist. Daher muss die meiste Arbeit manuell erledigt werden.

8.1 Wie Übersetzungen in Debian gehandhabt werden

Die Handhabung der Übersetzung von Texten, die in Paketen enthalten sind, ist immer noch eine manuelle Aufgabe und der Prozess hängt von der Art des Textes ab, den Sie übersetzt sehen wollen.

Meistens wird für Programm-Nachrichten die gettext-Infrastruktur verwendet. Dabei werden die Übersetzungen oft direkt in den jeweiligen Projekten gepflegt, wie bei dem Free Translation Project, bei dem GNOME Translation Project oder bei dem KDE Localization Projekt. Die einzige zentrale Ressource bei Debian ist die zentrale Übersetzungsstatistik von Debian, wo Statistiken zu übersetzen Dateien gefunden und heruntergeladen werden können.

Für Paketbeschreibungen gibt es seit vielen Jahren Übersetzungen und Betreuer müssen nichts besonders tun um Übersetzungen von Paketbeschreibungen zu unterstützen, Übersetzer für Paketbeschreibungen sollten das Debian Übersetzungsprojekt für Beschreibungen (DDTP) benutzen.

Für `debconf`-Vorlagen sollten Betreuer das `po-debconf`-Paket verwenden, um die Arbeit von Übersetzern zu vereinfachen. Einige Statistiken können auf der Seite zur zentralen Übersetzungsstatistik von Debian gefunden werden.

Für Webseiten hat jedes L10n-Team Zugriff auf das passende VCS und die Statistiken sind auf der Site für die zentrale Übersetzungsstatistik von Debian verfügbar.

Für allgemeine Dokumentation über Debian entspricht der Prozess mehr oder weniger dem der Webseiten (für Übersetzer, die Zugriff auf das VCS haben), aber es gibt dort keine Statistikseiten.

Ein weiterer Teil der Internationalisierungsarbeit betrifft paketspezifische Dokumentation (Handbuchseiten, Info-Dokumente und andere Formate). Zumindest die Handbuchseiten werden mit po-Dateien übersetzt, wie die meisten anderen bereits erwähnten Dateien.

8.2 I18N & L10N FAQ für Paketbetreuer

Dies ist eine Liste von Problemen, denen Betreuer bezüglich I18n und L10n gegenüberstehen. Behalten Sie, während Sie dies lesen, im Hinterkopf, dass es über diese Punkte innerhalb von Debian keine Einigkeit gibt und folgendes nur ein Ratschlag sein kann. Falls Sie für die vorliegenden Probleme bessere Ideen haben oder mit einigen Punkten nicht einverstanden sind, tun Sie sich keinen Zwang an und geben Sie Ihre Rückmeldung, so dass dieses Dokument verbessert werden kann.

8.2.1 Wie ein vorliegender Text übersetzt wird

Um Paketbeschreibungen zu übersetzen, müssen Sie nichts tun. Die DDTP-Infrastruktur wird das zu übersetzende Material an Freiwillige schicken, ohne dass Sie dazu etwas tun müssen.

Für alle weiteren Dokumente (`debconf`-Vorlagen, gettext-Dateien, Handbuchseiten oder andere Dokumentation) sollte eine Anfrage für die Übersetzung in weitere Sprachen an debian-i18n geschrieben werden. Einige Mitglieder der Übersetzungs-Teams stehen auf der Liste und werden die weitere Koordinierung übernehme, um die Materialien übersetzt und gesichtet zu bekommen. Wenn die Übersetzungen fertig sind, werden Sie die übersetzten Dokumente per E-Mail oder per `wishlist`-Fehlerbericht bekommen. Es wird empfohlen die `po-debconf`-Werkzeuge für die I18n-Integration zu verwenden.

8.2.2 Wie eine vorliegende Übersetzung überprüft wird

Von Zeit zu Zeit übersetzen Einzelpersonen einige Texte in Ihrem Paket und bitten Sie, die Übersetzung in das Paket aufzunehmen. Dies kann problematisch werden, falls Sie die vorliegende Sprache nicht fließend sprechen. Es ist eine gute Idee, das Dokument an die entsprechende L10n-Mailingliste zu senden und um eine Überprüfung zu bitten. Sobald dies erledigt ist, sollten Sie von der Qualität der Übersetzug überzeugt sein und sich sicherer fühlen, sie in Ihr Paket einzubinden.

8.2.3 Wie eine vorliegende Übersetzung aktualisiert wird

Falls Sie einige Übersetzungen eines vorliegendens Textes herumliegen haben, sollten Sie jedesmal, wenn Sie das Original aktualisieren, den letzten Übersetzer bitten, die Übersetzung mit Ihren neuen Änderungen zu aktualisieren. Behalten Sie im Hinterkopf, dass diese Aufgabe Zeit beansprucht; mindestens eine Woche, um die Aktualisierung korrekturlesen zu lassen und so weiter.

Falls der Übersetzer nicht reagiert, könnten Sie auf der entsprechenden L10n-Mailingliste um Hilfe ersuchen. Falls alles scheitert, vergessen Sie nicht, eine Warnung im übersetzten Dokument zu hinterlassen, die angibt, dass die Übersetzung veraltet ist und der Leser sich, wenn möglich, auf das Originaldokument beziehen sollte.

Vermeiden Sie es, eine Übersetzung vollständig zu entfernen, weil sie veraltet ist. Veraltete Dokumentation ist für Leute, die kein Englisch sprechen, oft besser als gar keine Dokumentation.

8.2.4 Wie Fehlerberichte gehandhabt werden, die eine Übersetzung betreffen

Die beste Lösung ist wahrscheinlich, den Fehler als zu den Originalautoren weitergeleitet zu kennzeichnen und ihn sowohl an den letzten Übersetzer als auch an sein Team zu senden (unter Benutzung der entsprechenden debian-l10n-XXX-Mailingliste).

8.3 I18n- & L10n-FAQ für Übersetzer

Behalten Sie, während Sie dies lesen, im Hinterkopf, dass es über diese Punkte innerhalb von Debian keine Einigkeit gibt, und dass Sie auf jeden Fall mit Ihrem Team und dem Paketbetreuer zusammenarbeiten sollten.

8.3.1 Wie man bei Übersetzungsbemühungen helfen kann

Wählen Sie aus, was Sie übersetzen möchten und stellen Sie sicher, dass nicht bereits jemand daran arbeitet (benutzen Sie Ihre debian-l10n-XXX-Mailingliste), übersetzen Sie es, lassen Sie es durch andere mit dieser Muttersprache auf Ihrer L10n-Mailingliste überprüfen und stellen Sie es dem Paketbetreuer zur Verfügung (siehe nächsten Punkt).

8.3.2 Wie eine Übersetzung zur Eingliederung in ein Paket bereitgestellt wird

Stellen Sie sicher, dass Ihre Übersetzung korrekt ist (bitten Sie auf Ihrer L10n-Mailingliste um eine Überprüfung), bevor Sie sie zur Eingliederung bereitstellen. Es wird für jeden eine Zeitersparnis sein und das Chaos vermeiden, das daraus resultiert, dass Sie mehrere Versionen des gleichen Dokuments in Fehlerberichten haben.

Die beste Lösung ist es, einen regulären Fehlerbericht gegen das Paket einzureichen, der die Übersetzung enthält. Stellen Sie sicher, dass Sie die Kennzeichnung "patch", "l10n" und keinen Schweregrad höher als "wishlist" verwenden, da das Fehlen einer Übersetzung ein Programm niemals an der Ausführung hindert.

8.4 Beste aktuelle Vorgehensweise bezüglich L10n

- Bearbeiten Sie als Betreuer niemals die Übersetzungen in irgendeiner Weise (auch nicht, um das Aussehen neu zu formatieren), ohne mit der entsprechenden L10n-Mailingliste zu sprechen. Sie riskieren zum Beispiel, die Zeichenkodierung der Datei zu zerstören, wenn Sie dies tun. Außerdem könnte das, was Sie als Fehler betrachten, in der vorliegenden Sprache richtig sein (oder sogar nötig).
- Falls Sie als Übersetzer einen Fehler im Originaltext finden, stellen Sie sicher, dass er gemeldet wird. Übersetzer sind oft die aufmerksamsten Leser eines vorliegenden Textes und falls Sie die gefundenen Fehler nicht melden, wird es niemand tun.
- Denken Sie auf jeden Fall daran, dass der Hauptstreitpunkt bei L10n darin besteht, dass mehrere Leute für eine Zusammenarbeit nötig sind und dass es sehr leicht ist, selbst für kleine Probleme aufgrund von Missverständnissen einen Flame-War (schwerwiegende Streitigkeit) heraufzubeschwören. Falls Sie daher Probleme mit einem Gesprächspartner haben, fragen Sie auf der entsprechenden L10n-Mailingliste, Debian-I18n oder sogar Debian-Devel nach Hilfe (aber Vorsicht, L10n-Diskussionen führen auf jener Liste oft zu Flame-Wars :)).
- Kooperation kann auf jeden Fall nur mit **gegenseitigem Respekt** erreicht werden.

KAPITEL 9

Überblick über die Werkzeuge der Debian-Betreuer

Dieser Abschnitt enthält eine grobe Übersicht über die Werkzeuge, die Betreuern zur Verfügung stehen. Das Folgende ist beileibe nicht vollständig oder maßgeblich, sondern nur eine Anleitung für einige der beliebstesten Werkzeuge.

Debian-Betreuerwerkzeuge sind dazu gedacht, Entwicklern zu helfen und Zeit für wirklich kritische Aufgaben nutzen zu können. Wie schon Larry Wall sagte, gibt es mehr als einen Weg, um etwas zu erledigen.

Einige Paktebetreuer bevorzugen die Benutzung von hochentwickelten Paketverwaltungswerkzeugen, andere wiederum nicht. Debian ist dies egal. Jedes Werkzeug ist gut was das diese Aufgabe erfüllt. Daher ist dieser Abschnitt nicht dazu gedacht, jemandem vorzuschreiben, welche Werkzeuge er benutzen oder wie er mit seinen Pflichten als Betreuer umgehen soll. Er ist auch nicht dazu gedacht, ein besonderes Werkzeug zu befürworten, um ein anderes auszuschließen.

Die meisten Beschreibungen dieser Pakete entstammen selbst den tatsächlichen Paketbeschreibungen. Weitere Informationen zu den Pakten sind in der Paketbeschreibung selbst zu finden. Sie können außerdem mit dem Befehl `apt-cache show`*Paketname* zusätzliche Informationen abrufen.

9.1 Hauptwerkzeuge

Die folgenden Werkzeuge werden größtenteils von jedem Betreuer benötigt.

9.1.1 dpkg-dev

`dpkg-dev` enthält die Werkzeuge (einschließlich `dpkg-source`), die benötigt werden, um Debian-Pakete zu entpacken, zu erstellen und hochzuladen. Diese Hilfswerkzeuge enthalten die entsprechend untergeordneten Funktionalitäten, die zum Erstellen und Manipulieren von Paketen benötigt werden; als solche sind diese für jeden Debian-Betreuer erforderlich.

9.1.2 debconf

`debconf` stellt eine einheitliche Schnittstelle zur Verfügung, um Pakete interaktiv konfigurieren zu können. Es gibt unterschiedliche Bedienoberflächen, d.h. es erlaubt Endanwendern, Pakete mit einer reinen Textoberfläche, einer HTML-Oberfläche oder einer Dialogoberfläche zu konfigurieren. Neue Bedienoberflächen können als Module hinzugefügt werden.

Dokumentation für dieses Paket ist im Paket `debconf-doc` enthalten.

Viele sind der Ansicht, dieses System sollte für alle Pakete verwendet werden, die eine interaktive Konfiguration erfordern; siehe *Konfigurationsverwaltung mit debconf*. Derzeit ist `debconf` nicht in den Debian-Richtlinien vorgeschrieben, was sich aber zukünftig ändern könnte.

9.1.3 fakeroot

`fakeroot` simuliert Root-Rechte. Dies ermöglicht Ihnen, Pakete zu erstellen, ohne Root zu sein (Pakete möchten üblicherweise Dateien mit Root-Besitzrechten installieren). Falls Sie `fakeroot` installiert haben, wird `dpkg-buildpackage` es automatisch benutzen.

9.2 Lint-Werkzeuge für Pakete

Gemäß dem Free On-line Dictionary of Computing (FOLDOC) ist `lint` ein Unix-Verarbeitungsprogramm für die Sprache C, das gründlichere Code-Prüfungen enthält als übliche C-Compiler. Lint-Werkzeuge für Pakete helfen Paketbetreuern, häufige Probleme und Richtlinienverletzungen in ihren Paketen automatisiert zu finden.

9.2.1 lintian

`lintian` zerlegt Debian-Pakete und gibt Informationen über Fehler und Richtlinien-Verletzungen aus. Es enthält automatisierte Prüfungen für viele Gesichtspunkte der Debian-Richtlinien, wie auch einige Prüfungen auf häufige Fehler.

Sie sollten sich regelmäßig das neueste `lintian` aus `Unstable` installieren und all Ihre Pakete überprüfen. Beachten Sie, dass die Option `-i` detaillierte Erklärungen liefert, was jeder Fehler oder jede Warnung bedeutet, was deren Grundlage in den Debian-Richtlinien ist und wie das Problem üblicherweise behoben werden kann.

Für weitere Informationen darüber wie und wann Lintian benutzt wird auf *Das Paket testen* verwiesen.

You can also see a summary of all problems reported by Lintian on your packages at https://udd.debian.org/lintian/. The reports contain the latest `lintian` output for the whole development distribution (`unstable`).

9.2.2 lintian-brush

`lintian-brush` enthält einen Satz an Skripten welche mehr als 80 bekannte Lintian Fehler in Debian Paketen automatisch korrigieren können.

Es kommt mit einem Wrapperskript welches die anderen Skripte aufruft, die Changelog Einträge erstellt (sofern gewünscht) und jede Änderung in die Versionskontrolle einträgt.

9.2.3 piuparts

`piuparts` ist ein Werkzeug was die `.deb` Paketinstallation, die Paketaktualisierung und das Entfernen eines Paketes testet.

`piuparts` testet das `.deb` Pakete die Installation, eine Aktualisierung und das Entfernen des Paketes korrekt ausgeführen. Dies wird durch die Installation eines Minimal Debians in einem Chroot erreicht indem wiederum die eigentliche Installation, das Upgrade und das Entfernen des Paketes ausgeführt wird und dabei den Zustand des Verzeichnisbaums am Ende der Tests mit dem Zustand vor den Tests vergleicht. `piuparts` meldet alle Dateien die hinzugefügt, entfernt oder während des Prozesses verändert worden sind.

`piuparts` ist als Qualitätsicherungswerkzeug für Personen gedacht welche `.deb` Pakete bauen und diese testen wollen bevor diese ins Debian Archive geladen werden.

9.2.4 debdiff

`debdiff` (aus dem Paket `devscripts`, *devscripts*) vergleicht die Dateilisten und "control"-Dateien zweier Pakete. Es ist ein einfacher Rückfalltest, der Ihnen hilft festzustellen, ob sich die Anzahl der Binärpakete seit dem letzten Upload verändert hat oder ob sich etwas in der "control"-Datei geändert hat. Natürlich werden einige Unterschiede, die ausgegeben werden, in Ordnung sein, aber es kann Ihnen helfen, verschiedene Missgeschicke zu vermeiden.

Sie können es für zusammengehörige Binärpakete ausführen:

```
debdiff package_1-1_arch.deb package_2-1_arch.deb
```

Oder auch für zusammengehörige "changes"-Dateien:

```
debdiff package_1-1_arch.changes package_2-1_arch.changes
```

Um weitere Informationen zu erhalten, lesen Sie debdiff 1.

9.2.5 diffoscope

`diffoscope` bietet einen detaillierten Vergleich von Dateien, Archiven und Verzeichnissen.

`diffoscope` versucht den Unterschieden zwischen Dateien oder Verzeichnissen auf den Grund zu gehen. Es entpackt rekursiv Archive vieler Art und wandelt verschiedene Binärformate in eine besser lesbare Form um, um sie zu vergleichen.

Ursprünglich wurde es entwickelt um `.deb` Dateien oder zwei `changes` Dateien zu vergleichen, inzwischen kann es zwei Tarballs, ISO-Images oder PDF Dateien genauso einfach vergleichen und unterstützt eine Vielzahl von Dateitypen.

Die Unterschiede können als Text, als HTML Report oder JSON ausgegeben werden.

9.2.6 duck

`duck`, der Debian Url CheckKer, verarbeitet mehrere Felder in den Dateien `debian/control`, `debian/upstream`, `debian/copyright`, `debian/patches/*`, aber auch in Systemd Unit Dateien `systemd.unit`, es prüft dabei ob URLs, VCS Verlinkungen und E-Mail Adressen-Domainen, die gefunden worden sind, noch gültig sind.

9.2.7 adequate

`adequate` prüft Pakete, die aktuell auf dem System installiert sind, und meldet Regelverletzungen.

Die folgenden Tests sind aktuell implementiert:

- defekte Symlinks
- fehlende Copyright Datei
- obsolete Konfigurationsdateien
- Python Module die nicht Byte-kompiliert sind
- fehlende Bibliotheken, unbekannte Symbole, nicht passende Größen bei Symbolen
- Kollisionen von Programmnamen
- fehlende Alternativen
- fehlende `binfmt` Interpreter und Detektoren
- fehlende `pkg-config` Abhängigkeiten
- ungültige Nutzer-/Gruppen-Werte in systemd-/D-Bus-/sysvinit-Dateien.

Siehe /usr/share/doc/adequate/README wie `adequate` automatisch durch `autopkgtest` ausgeführt werden kann.

9.2.8 i18nspector

`i18nspector` ist ein Werkzeug welches Dateien für Übersetzungstemplates (POT), Nachrichtenkataloge (PO) und kompilierte Kataloge (MO) auf gängige Probleme prüft.

9.2.9 cme

cme is a configuration editor. With libconfig-model-dpkg-perl installed, it can edit and validate Debian source package files.

9.2.10 licensecheck

licensecheck versucht die Lizenz für jede an sie übergebene Datei zu ermitteln, indem nach Textpassagen im Kopf der Dateien gesucht wird die zu verschiedenen Lizenzen gehören.

9.2.11 blhc

blhc ist ein Werkzeug welches in den Logdateien nach dem Bau der Pakete nach fehlenden Hardening Flags sucht.

9.3 Helferskripte für debian/rules

Paketerstellungswerkzeuge erleichtern das Verfassen von debian/rules-Dateien. Lesen Sie *Helferskripte*, um weitere Informationen darüber zu erhalten, warum dies erwünscht und jenes unerwünscht sein könnte.

9.3.1 debhelper

debhelper ist eine Programmsammlung, die in debian/rules benutzt werden kann, um häufige Aufgaben zu automatisieren, die sich auf das Erstellen binärer Debian-Pakete beziehen. debhelper enthält Programme, um verschiedene Dateien in Ihre Pakete zu installieren, Dateien zu komprimieren, Dateirechte zu korrigieren und Ihr Paket in das Debian-Menüsystem zu integrieren.

Anders als bei anderen Lösungen ist debhelper in mehrere kleine, einfache Befehle aufgeteilt, die auf eine durchgängige Art zusammenarbeiten und dabei eine verglichen mit anderen Tools fein granuläre Kontrolle für debian/rules erlaubt.

Es gibt eine zu große Zahl kleiner Erweiterungspakete für debhelper, die zu verä nderlich sind, um sie hier zu dokumentieren. Sie können die Liste der meisten von Ihnen ansehen, indem Sie apt-cache search ^dh- aufrufen.

Wenn Sie den debhelper Kompatibilitätslevel für ihr Paket wählen, sollten Sie die höchste Kompatibilitätsstufe auswählen, die in der neuesten stabilen Version unterstützt wird. Verwenden Sie eine höhere Kompatibilitätsstufe nur, wenn Sie bestimmte Funktionen benötigen, die von dieser Kompatibilitätsstufe bereitgestellt werden und in früheren Stufen nicht verfügbar sind.

In der Vergangenheit wurde die Kompatibilitätsstufe in debian/compat definiert. Heutzutage ist es jedoch besser, dies nicht mehr zu verwenden, sondern eine versionierte Build-Abhängigkeit wie z.B. debhelper-compat (= 12) zu verwenden.

9.3.2 dh-make

Das Paket dh-make enthält ein Programm namens dh_make, das ein Gerüst von Dateien erstellt, die nötig sind, um Debian-Pakete aus einem Quellcode-Verzeichnisbaum zu erstellen. Wie der Name schon nahelegt, ist dh_make eine Neufassung von debmake, dessen Vorlagendateien dh_*-Programme von debhelper benutzen.

Während die von dh_make generierten "rules"-Dateien im Allgemeinen eine ausreichende Basis für ein funktionierendes Paket bilden, gibt es trotzdem noch weitere grundlegende Arbeiten zu erledigen: Die Hauptarbeit für die Feinabstimmung und das Paket funktional und Richtlinien konform zu machen, liegt immer noch beim Paketbetreuer.

9.3.3 equivs

equivs ist ein weiteres Paket für die Paketerstellung. Es wird oft für den lokalen Gebrauch vorgeschlagen, falls Sie einfach ein Paket erstellen müssen, um Abhängigkeiten zu erfüllen. Es wird manchmal auch benutzt, um *Meta-Pakete* zu erstellen. Dabei handelt es sich um Pakete, deren einziger Zweck darin besteht, Abhängigkeiten zu anderen Paketen zu generieren.

9.4 Werkzeuge zum Paketbau

Die folgenden Pakete helfen beim Prozess der Paketerstellung und führen im Allgemeinen `dpkg-buildpackage` aus, um unterstützende Aufgaben zu behandeln.

9.4.1 git-buildpackage

`git-buildpackage` stellt die Fähigkeit bereit, Debian-Quellpakete in ein Git-Repository aufzunehmen oder zu importieren, ein Debian-Paket aus dem Git-Repository zu bauen und bei der Integration von Änderungen der Originalautoren in das Repository zu helfen.

Diese Hilfswerkzeuge bieten eine Infrastruktur, um Debian-Betreuern den Gebrauch von Git zu erleichtern. Dies ermöglicht es, getrennte Git-Zweige von Paketen für die Distributionen `Stable`, `Unstable` und möglicherweise `Experimental` vor zuhalten, zusammen mit den anderen Vorteilen eines Versionsverwaltungssystems.

9.4.2 debootstrap

Das Paket und Skript `debootstrap` ermöglicht Ihnen das Urladen eines Debian-Basissystems in irgendeinen Teil Ihres Dateisystems. Mit Basissystem ist hier das Minimum an installierten Paketen gemeint, die nötig sind, um den Rest des Systems zu betreiben und zu installieren.

Ein solches System zu haben, kann in vielerlei Hinsicht nützlich sein. Sie können zum Beispiel mit `chroot` in das System gehen und Ihre Build-Abhängigkeiten testen. Oder Sie können testen, wie sich Ihr Paket verhält, wenn es in ein nacktes Basissystem installiert wird. Chroot-Builder benutzen dieses Paket; siehe weiter unten.

9.4.3 pbuilder

`pbuilder` erstellt ein Chroot-System und baut ein Paket innerhalb der Chroot-Umgebung. Es ist sehr nützlich, um die Build-Abhängigkeiten des Pakets zu überprüfen und sicherzustellen, dass keine unnötigen oder falschen Build-Abhängigkeiten in dem resultierenden Paket existieren.

Ein verwandtes Paket ist `cowbuilder`, das den Bauprozess mittels eines COW-Dateisystems auf jedem Standard-Linux-Dateisystem beschleunigt.

9.4.4 sbuild

`sbuild` ist ein weiterer automatisierter Builder. Er kann auch Chroot-Umgebungen benutzen. Er kann eigenständig benutzt werden oder als Teil einer verteilten Build-Umgebung über ein Netzwerk. Als letzteres ist er Teil des Systems, das Portierer benutzen, um Binärpakete für all die verfügbaren Architekturen zu erstellen. Weitere Informationen finden Sie in *wanna-build* und unter https://buildd.debian.org/ können Sie das System in Aktion sehen.

9.5 Programme zum Hochladen von Paketen

Die folgenden Pakete helfen, den Prozess, Pakete in das offizielle Archiv hochzuladen, zu automatisieren oder zu vereinfachen.

9.5.1 dupload

Das Paket `dupload` enthält ein gleichnamiges Skript mit dem automatisierte Dinge ausgeführt werden können wie ein Paket ins Debian Archiv hochzuladen, dieses Hochladen zu monitoren und optional eine E-Mail nach einen Paket Upload zu verschicken. Es unterstützt verschiedene Arten von Hooks um die eigene Funktionalität zu erweitern, es kann aber auch so konfiguriert werden, dass es neue Uploadziele oder Methoden unterstützt. Standardmäßig beinhaltet es schon verschiedene Hooks die Tests ausführen und beinhaltet ebenso eine Konfiguration die alle Uploadziele innerhalb Debians unterstützt.

9.5.2 dput

Mit dem Paket und Skript `dput` lässt sich das gleiche erreichen wie mit `dupload`, aber auf eine andere Art. Standardmäßig wird die Ausführung von `dinstall` im Leerlaufmodus (dry-run) nach dem Hochladen unterstützt.

9.5.3 dcut

Das Skript `dcut` (Teil des Pakets `dput`, *dput*) hilft beim Entfernen von Dateien aus dem FTP-Upload-Verzeichnis.

9.6 Automatisierung der Verwaltungsaufgaben

Die folgenden Werkzeuge helfen, verschiedene Verwaltungsaufgaben vom Hinzufügen von Änderungsprotokolleinträgen oder Signaturzeilen über das Nachschlagen von Fehlern in Emacs bis zum Gebrauch vom neuesten und offiziellen `config.sub` zu automatisieren.

9.6.1 devscripts

`devscripts` ist ein Paket mit Wrappern und Werkzeugen, die für die Pflege Ihrer Debian-Pakete sehr hilfreich sind. Beispielskripte sind `debchange` (oder sein Alias `dch`), das Ihre `debian/changelog`-Datei über die Befehlszeile bearbeitet, bis hin zu `debuild` was ein ein Wrapperskript um `dpkg-buildpackage` herum ist. Das Dienstprogramm `bts` ist auch sehr hilfreich, um z.B. den Status von Fehlerberichten von der Kommandozeile aus zu aktualisieren. `uscan` kann verwendet werden, um nach neuen Upstream-Versionen Ihrer Pakete zu suchen (Siehe hierzu https://wiki.debian.org/debian/watch für mehr Informationen). `suspicious-source` gibt eine Liste von Dateien aus, die keine allgemein bekannten Quelldateien sind.

Eine vollständige Liste der verfügbaren Skripte finden Sie auf der Handbuchseite devscripts(7).

9.6.2 reportbug

`reportbug` ist ein Werkzeug welches geschaffen wurde um das Melden von Fehlern in Debian und abgeleiteten Derivaten relativ problemlos zu gestalten. Zu den Eigenschaften gehören:

- Integration mit mutt und mh/nmh E-Mail Readern.
- Zugriff auf offene Fehlerberichte, um leichter feststellen zu können, ob bereits Probleme gemeldet wurden.
- Automatisches prüfen nach neuen Versionen von Paketen.

`reportbug` ist für die Verwendung auf Systemen mit einem installierten MTA (Mail Transport Agent) vorgesehen. Sie können die Konfigurationsdatei jedoch bearbeiten und Berichte über einen beliebigen verfügbaren Mailserver senden.

Dieses Paket enthält auch das Skript `querybts` um das Debian Bug Tracking System durchsuchen zu können.

9.6.3 autotools-dev

`autotools-dev` enthält optimale Vorgehensweisen für Maintainer, die Pakete betreuen, in denen `autoconf` und/oder `automake` zum Einsatz kommt. Außerdem enthält es anerkannte `config.sub` und `config.guess` Dateien, von denen bekannt ist, dass sie auf allen Debian-Portierungen funktionieren.

9.6.4 dpkg-repack

`dpkg-repack` erstellt eine Debian-Paketdatei aus einem Paket, das bereits installiert wurde. Falls irgendwelche Änderungen vorgenommen wurden, während das Paket entpackt war (es wurden z.B. Dateien in `/etc` verändert), wird das neue Paket die Änderungen erben.

Dieses Hilfswerkzeug kann das Kopieren von Paketen von einem Rechner zu einem anderen, das Neuerstellen von Paketen, die auf Ihrem System installiert wurden, aber nirgendwo mehr verfügbar sind, oder das Sichern des derzeitigen Paketstatus vor dem Upgrade vereinfachen.

9.6.5 alien

`alien` konvertiert Binärpakete zwischen verschiedenen Paketformaten, einschließlich Debian, RPM (RedHat), LSB (Linux Standard Base), Solaris und Slackware.

9.6.6 elpa-dpkg-dev-el

`elpa-dpkg-dev-el` is an Emacs lisp package that provides assistance when editing some of the files in the `debian` directory of your package. For instance, there are handy functions for listing a package's current bugs, and for finalizing the latest entry in a `debian/changelog` file.

9.6.7 dpkg-depcheck

`dpkg-depcheck` (aus dem Paket `devscripts`, *devscripts*) führt einen Befehl unter `strace` aus, um festzustellen, welche Pakete vom angegebenen Befehl benutzt werden.

Für Debian-Pakete ist dies nützlich, wenn Sie eine `Build-Depends`-Zeile für Ihr neues Paket erstellen müssen: Wenn Sie den Build-Prozess durch `dpkg-depcheck` auszuführen, dann wird eine gute erste Übersicht über die Build-Abhängigkeiten ausgegeben werden. Zum Beispiel so aufgerufen:

```
dpkg-depcheck -b debian/rules build
```

`dpkg-depcheck` kann außerdem benutzt werden, um Laufzeitabhängigkeiten zu prüfen, insbesondere, wenn Ihr Paket exec 2 benutzt, um andere Programme auszuführen.

Weitere Informationen finden Sie unter dpkg-depcheck 1.

9.6.8 debputy

Das Programm `debputy` ist seit 2024 neu. Während der Hauptzweck des Programms darin besteht ein neues Build-Paradigma für Debian-Pakete bereitzustellen, enthält `debputy` einige Befehle, die mit jedem bestehenden Debian-Paket verwendet werden können, um die Korrektheit der meisten Dateien in `debian/*` zu prüfen und in vielen Fällen auch automatisch herstellen zu können.

Um die Korrektheit von Dateien in debian/* zu überprüfen, kann dieser Befehl verwendet werden:

```
debputy lint --spellcheck
```

Die Dateien debian/control und debian/tests/control können so formatiert werden:

```
debputy reformat --style black
```

Die Verwendung des Befehls `reformat` ersetzt die Verwendung von `wrap-and-sort -ast`.

Das Werkzeug `debputy` enthält ebenfalls einen Sprach-Server, der, wenn in einen Editor integriert, Echtzeit-Rückmeldung über die Korrektheit von Dateien in `debian/*` während der Bearbeitung gibt.

Um weitere Informationen zu erhalten, lesen Sie debputy 1.

9.7 Portierungswerkzeuge

Die folgenden Werkzeuge sind hilfreich für Portierer und zur Kompilierung für andere Plattformen.

9.7.1 dpkg-cross

`dpkg-cross` ist ein Werkzeug, um Bibliotheken und Header zum Kompilieren auf anderen Plattformen auf eine Art zu installieren, die `dpkg` ähnlich ist. Weiterhin verbessert es die Funktionalität von `dpkg-buildpackage` und `dpkg-shlibdeps`, um das Kompilieren von Paketen für andere Plattformen (cross-compiling) zu unterstützen.

9.8 Dokumentation und Information

Die folgenden Pakete stellen Informationen für Betreuer zur Verfügung oder helfen bei der Erstellung von Dokumentation.

9.8.1 debian-policy

Das Paket `debian-policy` enthält das Debian Richtlinienhandbuch und zugehörige Dokumente, diese sind:

- Debian Richtlinienhandbuch
- Filesystem Hierarchy Standard (FHS)
- Regeln für Menüs in Debian
- Spezifische Regeln für Perl in Debian
- Spezifikationen des Debian Konfigurationsmanagment
- Spezifikation für eine maschinell lesbare Datei debian/copyright
- Autopkgtest - Automatisches Testen von Paketen im installierten Zustand
- Maßgebliche Liste der Namen von virtueller Paketen
- Checkliste mit Regeln für das Aktualisieren Ihrer Paketen

Das Debian-Richtlinienhandbuch enthält die Richtlinien zu Paketen und Einzelheiten zum Mechanismus des Paketierens. Es deckt alles ab, von den erforderlichen "gcc"-Optionen bis hin zur Funktionsweise der Betreuerskripte ("postinst" usw.), Sektionen für Paketen und Prioritäten usw.

Auch sehr nützlich ist die Datei `/usr/share/doc/debian-policy/upgrading-checklist.txt.gz`, diese listet die Änderungen zwischen den verschieden Versionen der Debian Richtlinie auf.

9.8.2 doc-debian

`doc-debian` enthält ein große Menge an Debian spezifischer Dokumentation:

- Debian Linux Manifesto
- Verfassung des Debian Projektes
- Debian Gesellschaftervertrag
- Debian Free Software Guidelines
- Dokumentation des Debian Bug Tracking System
- Einführung in die Debian Mailinglisten

9.8.3 developers-reference

Das Paket `developers-reference` enthält das Dokument, welches Sie gerade lesen, die Debian Developer's Reference ist eine Reihe von Richtlinien und Best Practices, die von und für die Gemeinschaft der Debian-Entwickler erstellt wurden.

9.8.4 maint-guide

Das Paket `maint-guide` enthält den "Debian-Leitfaden für Neue Paketbetreuer".

Dieses Dokument versucht den normalen Debian-Benutzern und potenziellen Entwicklern den Aufbau eines Debian-Pakets zu beschreiben. Es verwendet bewusst eine nicht-technische Sprache und hat ein Reihe an Arbeitsbeispielen.

9.8.5 debmake-doc

Das Paket debmake-doc enthält den "Leitfaden für Debian-Betreuer".

Dieses Dokument ist neuer als der "Debian-Leitfaden für Neue Paketbetreuer" und zielt darauf ab, es zu ersetzen. Der "Leitfaden für Debian-Betreuer" richtet sich an Leute, die Paketierung für Debian erlernen wollen, und deckt eine breite Spanne von Themen und Werkzeugen ab, zusammen mit vielen Beispielen zu verschiedenen Paketierungsproblemen.

9.8.6 packaging-tutorial

Dieses Tutorial ist eine Einführung in die Paketierung von Debian Paketen. Hier erfahren angehende Entwickler, wie sie vorhandene Pakete ändern, eigene Pakete erstellen können und mit der Debian-Gemeinschaft interagieren können.

Zusätzlich zum Haupt-Tutorial enthält es drei praktische Beispiele wie zum ändern des grep Paketes, zum Paketieren des gnujump Spiels und einer Java Bibliothek.

9.8.7 how-can-i-help

how-can-i-help zeigt Möglichkeiten für Beiträge zu Debian auf. how-can-i-help hängt sich in APT, um Möglichkeiten für Beiträge zu Debian (verwaiste Pakete, Fehler mit dem Tag 'newcomer') für lokal installierte Pakete nach jedem APT Aufruf aufzulisten. Es kann auch direkt aufgerufen werden und listet dann alle Beitragsmöglichkeiten auf (nicht nur die neuen).

9.8.8 docbook-xml

docbook-xml stellt die DocBook-XML-Dokumenttypdefinitionen (DTD) bereit, die häufig für Debian-Dokumentation benutzt werden (genauso wie die ältere Debiandoc-SGML-DTD).

Das Paket docbook-xsl stellt XSL-Dateien zum Erstellen und Gestalten der Datenquelle in verschiedenen Ausgabeformaten bereit. Sie benötigen einen XSLT-Prozessor wie xsltproc, um die XSL-Stylesheets zu verwenden. Dokumentation für die Stylesheets finden Sie in den verschiedenen docbook-xsl-doc-*-Paketen.

To produce PDF from FO, you need an FO processor, such fop or dblatex.

9.8.9 debiandoc-sgml

debiandoc-sgml stellt die DebianDoc-SGML-Dokumenttypdefinitionen (DTD) bereit, die normalerweise für Debian-Dokumentation benutzt worden ist, aber nun als veraltet gilt (stattdessen sollte docbook-xml oder python3-sphinx benutzt werden).

9.8.10 debian-keyring

Dieses Paket enthält die öffentlichen OpenPGP-Schlüssel der Debian-Entwickler und Paketbetreuer. Siehe *Verwalten Ihres öffentlichen Schlüssels* und die Paketdokumentation für weitere Informationen.

9.8.11 elpa-debian-el

elpa-debian-el provides an Emacs mode for viewing Debian binary packages. This lets you examine a package without unpacking it.