Debian Developer's Reference

Выпуск 14.18

Developer's Reference Team

2026-11-02

Оглавление

1 Границы данного документа **3**

2 Applying to Become a Member **5**
2.1 Приступаем к работе . . . 5
2.2 Наставники и поручители Debian . . . 6
2.3 Registering as a Debian member . . . 6

3 Обязанности разработчика Debian **9**
3.1 Обязанности сопровождающего пакетов . . . 9
3.1.1 Работа по подготовке следующего `стабильного` выпуска . . . 9
3.1.2 Сопровождение пакетов в `стабильном` выпуске . . . 9
3.1.3 Работа с критичными для выпуска ошибками . . . 10
3.1.4 Координация с разработчиками основной ветки . . . 10
3.2 Административные обязанности . . . 11
3.2.1 Сопровождение вашей связанной с Debian информации . . . 11
3.2.2 Сопровождение вашего открытого ключа . . . 11
3.2.3 Голосование . . . 11
3.2.4 Decision making . . . 12
3.2.5 Вежливый уход в отпуск . . . 12
3.2.6 Уход в отставку . . . 12
3.2.7 Возвращение после ухода . . . 13

4 Resources for Debian Members **15**
4.1 Списки рассылки . . . 15
4.1.1 Базовые правила . . . 15
4.1.2 Базовые списки рассылки . . . 15
4.1.3 Специальные списки рассылки . . . 16
4.1.4 Запрос новых списков рассылки, связанных с разработкой . . . 16
4.2 Каналы IRC . . . 16
4.3 Документация . . . 17
4.4 Машины Debian . . . 17
4.4.1 Сервер bugs . . . 18
4.4.2 Сервер ftp-master . . . 18
4.4.3 Сервер www-master . . . 18
4.4.4 Веб-сервер people . . . 18
4.4.5 salsa.debian.org: Git repositories and collaborative development platform . . . 18
4.4.6 GitHub.com: Submitting pull requests to upstream repositories . . . 19
4.5 База данных разработчиков . . . 19
4.6 Архив Debian . . . 19
4.6.1 Разделы . . . 21
4.6.2 Архитектуры . . . 22

4.6.3 Пакеты . . . 22
4.6.4 Выпуски . . . 22
4.6.4.1 Стабильный, тестируемые и нестабильный выпуски . . . 23
4.6.4.2 Дополнительная информация о тестируемом выпуске . . . 23
4.6.4.3 Экспериментальный выпуск . . . 23
4.6.5 Кодовые имена выпусков . . . 24
4.7 Зеркала Debian . . . 25
4.8 Система входящих пакетов . . . 25
4.9 Информация о пакете . . . 26
4.9.1 В веб . . . 26
4.9.2 Утилита `dak ls` . . . 26
4.10 Система отслеживания пакетов Debian . . . 26
4.11 Обзор пакетов разработчика . . . 27
4.12 Установка Debian FusionForge: Alioth . . . 27
4.13 Goodies for Debian Members . . . 27

5 Управление пакетами **29**
5.1 Новые пакеты . . . 29
5.2 Запись изменений в пакете . . . 30
5.3 Тестирование пакета . . . 30
5.4 Схема пакета с исходным кодом . . . 31
5.5 Выбор выпуска . . . 32
5.5.1 Специальный случай: загрузка в `стабильный` и `предыдущий стабильный` выпуски . . . 32
5.5.2 Special case: the `stable-updates` suite . . . 33
5.5.3 Специальный случай: загрузка в `testing/testing-proposed-updates` . . . 33
5.6 Загрузка пакета . . . 33
5.6.1 Source and binary uploads . . . 33
5.6.2 Загрузка на `ftp-master` . . . 34
5.6.3 Задержанные загрузки . . . 34
5.6.4 Загрузки безопасности . . . 35
5.6.5 Другие очереди загрузки . . . 35
5.6.6 Notifications . . . 35
5.7 Определение раздела для пакета, подраздела и приоритета . . . 35
5.8 New upstream versions . . . 36
5.9 Работа с ошибками . . . 36
5.9.1 Мониторинг ошибок . . . 37
5.9.2 Ответ на ошибки . . . 37
5.9.3 Bug housekeeping . . . 37
5.9.4 Когда ошибки исправляются путём новых загрузок . . . 39
5.9.5 Работа с ошибками, связанными с безопасностью . . . 40
5.9.5.1 Debian Security Tracker . . . 40
5.9.5.2 Конфиденциальность . . . 41
5.9.5.3 Рекомендации по безопасности . . . 41
5.9.5.4 Подготовка пакетов для решения проблем безопасности . . . 42
5.9.5.5 Загрузка исправленного пакета . . . 43
5.10 Subscribing to package updates . . . 43
5.11 Перемещение, удаление, переименование, придание статуса осиротевшего, усыновление и повторное введение пакетов . . . 44
5.11.1 Перемещение пакетов . . . 44
5.11.2 Удаление пакетов . . . 44
5.11.2.1 Удаление пакетов из каталога `Incoming` (каталога входящих пакетов) 45
5.11.3 Замена или переименование пакетов . . . 45
5.11.4 Придание статуса осиротевшего пакета . . . 46
5.11.5 Усыновление пакета . . . 46
5.11.6 Повторное добавление пакетов . . . 46
5.12 Работа на переносом и перенос пакетов . . . 47
5.12.1 Будьте добры к тем, кто занимается переносом . . . 47

5.12.2 Руководство для загрузок теми, кто занимается переносом . . . 48
5.12.2.1 Повторная компиляция или только двоичные NMU . . . 49
5.12.2.2 Когда следует делать NMU, если вы занимаетесь переносом . . . 49
5.12.3 Инфраструктура переноса и автоматизация . . . 50
5.12.3.1 Списки рассылки и веб-страницы . . . 50
5.12.3.2 Инструменты переноса . . . 50
5.12.3.3 `wanna-build` . . . 50
5.12.4 Если ваш пакет *не* может быть перенесён . . . 51
5.12.5 Отмечаем несвободные пакеты как собираемые автоматически (auto-buildable) 51
5.13 Загрузки не-сопровождающим (NMU) . . . 52
5.13.1 Когда и как делать NMU . . . 52
5.13.2 NMU и файл `debian/changelog` . . . 53
5.13.3 Использование очереди `DELAYED/` . . . 54
5.13.4 NMU с точки зрения сопровождающего . . . 54
5.13.5 NMU исходного кода и двоичные NMU (binNMU) . . . 55
5.13.6 NMU и загрузки командой контроля качества . . . 55
5.13.7 NMU и командные загрузки . . . 55
5.14 Package Salvaging . . . 55
5.14.1 When a package is eligible for package salvaging . . . 56
5.14.2 How to salvage a package . . . 56
5.15 Совместное сопровождение . . . 57
5.16 Тестируемый выпуск . . . 58
5.16.1 Основы . . . 58
5.16.2 Обновления из нестабильного выпуска . . . 58
5.16.2.1 Устаревание . . . 59
5.16.2.2 Удаление из тестируемого выпуска . . . 59
5.16.2.3 Круговые зависимости . . . 60
5.16.2.4 Влияние пакета в тестируемом выпуске . . . 60
5.16.2.5 Подробности . . . 60
5.16.3 Прямые обновления тестируемого выпуска . . . 61
5.16.4 Часто задаваемые вопросы . . . 61
5.16.4.1 Что такое критичные для выпуска ошибки, как производится их подсчёт? . . . 61
5.16.4.2 Как установка какого-то пакета в `тестируемый` выпуск может сломать другие пакеты? . . . 61
5.17 The Stable backports archive . . . 62
5.17.1 Основы . . . 62
5.17.2 Exception to the testing-first rule . . . 62
5.17.3 Who can maintain packages in the stable-backports archive? . . . 62
5.17.4 When can one start uploading to stable-backports? . . . 62
5.17.5 How long must a package be maintained when uploaded to stable-backports? . . . 62
5.17.6 How often shall one upload to stable-backports? . . . 63
5.17.7 How can one learn more about backporting? . . . 63

6 Лучшие практики создания пакетов 65
6.1 Лучшие практики для `debian/rules` . . . 65
6.1.1 Сценарии-помощники . . . 65
6.1.2 Множественные двоичные пакеты . . . 66
6.2 Лучшие практики для `debian/control` . . . 66
6.2.1 The package name . . . 66
6.2.2 Общие принципы описания пакетов . . . 66
6.2.3 Резюме пакета или короткое описание . . . 67
6.2.4 Длинное описание . . . 68
6.2.5 Домашняя страница основной ветки разработки . . . 68
6.2.6 Размещение системы контроля версий . . . 68
6.2.6.1 Vcs-Browser . . . 69
6.2.6.2 Vcs-* . . . 69
6.3 Лучшие практики для `debian/changelog` . . . 69

6.3.1 Написание полезных пунктов файла изменений . . . 69
6.3.2 Selecting the upload urgency . . . 70
6.3.3 Распространённые неправильные представления о записях об изменениях . . . 70
6.3.4 Общие ошибки в записях об изменениях . . . 71
6.3.5 Дополнение журналов файлами `NEWS.Debian` . . . 71
6.4 Best practices around security . . . 72
6.5 Лучшие практики для сценариев сопровождающих . . . 72
6.5.1 Generic guidelines . . . 72
6.5.2 Expected behavior of complex postinst scripts . . . 73
6.6 Управление настройкой с помощью `debconf` . . . 73
6.6.1 Не злоупотребляйте debconf . . . 73
6.6.2 Общие рекомендации для авторов и переводчиков . . . 73
6.6.2.1 Пишите на правильном английском . . . 73
6.6.2.2 Будьте добры к переводчикам . . . 74
6.6.2.3 Отменяйте статус неясных строк когда исправляете опечатки или орфографию . . . 74
6.6.2.4 Не допускайте ничего по поводу интерфейсов . . . 75
6.6.2.5 Не используйте первое лицо . . . 75
6.6.2.6 Будьте нейтральны в гендерном отношении . . . 75
6.6.3 Определение полей шаблонов . . . 76
6.6.3.1 Тип . . . 76
6.6.3.2 Описание: краткое и расширенное описания . . . 77
6.6.3.3 Choices . . . 77
6.6.3.4 Default . . . 77
6.6.4 Template fields specific style guide . . . 77
6.6.4.1 Тип поля . . . 77
6.6.4.2 Поле Description . . . 78
6.6.4.3 Поле Choices . . . 78
6.6.4.4 Поле Default . . . 78
6.7 Интернационализация . . . 79
6.7.1 Обработка переводов debconf . . . 79
6.7.2 Интернационализованная документация . . . 79
6.8 Best practices for `debian/patches` . . . 80
6.9 Общие ситуации при создании пакетов . . . 80
6.9.1 Пакеты, использующие `autoconf`/`automake` . . . 80
6.9.2 Библиотеки . . . 80
6.9.3 Документация . . . 81
6.9.4 Конкретные типы пакетов . . . 81
6.9.5 Независящие от архитектуры данные . . . 82
6.9.6 Необходимость в конкретной локали во время сборки . . . 82
6.9.7 Делаем так, чтобы deborphan определят переходные пакеты . . . 82
6.9.8 Лучшие практики для файлов `.orig.tar.{gz,bz2,xz}` . . . 82
6.9.8.1 Чистый исходный код . . . 83
6.9.8.2 Повторная упаковка исходного кода основной ветки . . . 83
6.9.8.3 Изменение двоичных файлов . . . 84
6.9.9 Лучшие практики для отладочных пакетов . . . 84
6.9.9.1 Automatically generated debug packages . . . 84
6.9.9.2 Manual -dbg packages . . . 84
6.9.10 Лучшие практики для метапакетов . . . 85

7 Помимо создания пакетов **87**
7.1 Отправка отчётов об ошибках . . . 87
7.1.1 Отправка множества отчётов об ошибках за один раз (массовое заполнение отчётов об ошибках) . . . 88
7.1.1.1 Пользовательские метки . . . 88
7.2 Работа по контролю качества . . . 89
7.2.1 Ежедневная работа . . . 89
7.2.2 Вечеринки по исправлению ошибок . . . 89

7.3 Связь с другими сопровождающими . . . 89
7.4 Работа с неактивными и/или недоступными сопровождающими . . . 89
7.5 Dealing with repeatedly negligent Debian developers . . . 91
7.6 Взаимодействие с будущими разработчиками Debian . . . 91
7.6.1 Поручение пакетов . . . 91
7.6.1.1 Поручительство нового пакета . . . 92
7.6.1.2 Поручение обновления существующего пакета . . . 93
7.6.2 Granting upload permissions to DMs . . . 93
7.6.3 Поддержка новых разработчиков . . . 94
7.6.4 Обработка заявок новых сопровождающих . . . 94

8 Интернационализация и переводы 95
8.1 Как переводы обрабатываются в Debian . . . 95
8.2 ЧаВО по I18N и L10N для сопровождающих . . . 96
8.2.1 Как перевести некоторый данный текст . . . 96
8.2.2 Как проверить перевод . . . 96
8.2.3 Как обновить перевод . . . 96
8.2.4 Как работать с отчётом об ошибке, касающемся перевода . . . 96
8.3 ЧаВО по I18N и L10N для переводчиков . . . 97
8.3.1 Как помочь с переводом . . . 97
8.3.2 Как предоставить перевод для добавления его в пакет . . . 97
8.4 Лучшие текущие практики, касающиеся локализации . . . 97

9 Обзор инструментов Debian для сопровождающего 99
9.1 Базовые инструменты . . . 99
9.1.1 `dpkg-dev` . . . 99
9.1.2 `debconf` . . . 99
9.1.3 `fakeroot` . . . 100
9.2 Инструменты для проверки пакетов на предмет ошибок и соответствия стандартам . . . 100
9.2.1 `lintian` . . . 100
9.2.2 `lintian-brush` . . . 100
9.2.3 `piuparts` . . . 100
9.2.4 `debdiff` . . . 101
9.2.5 `diffoscope` . . . 101
9.2.6 `duck` . . . 101
9.2.7 `adequate` . . . 101
9.2.8 `i18nspector` . . . 102
9.2.9 `cme` . . . 102
9.2.10 `licensecheck` . . . 102
9.2.11 `blhc` . . . 102
9.3 Помощники для `debian/rules` . . . 102
9.3.1 `debhelper` . . . 102
9.3.2 `dh-make` . . . 102
9.3.3 `equivs` . . . 103
9.4 Сборщики пакетов . . . 103
9.4.1 `git-buildpackage` . . . 103
9.4.2 `debootstrap` . . . 103
9.4.3 `pbuilder` . . . 103
9.4.4 `sbuild` . . . 103
9.5 ПО для загрузки пакетов . . . 103
9.5.1 `dupload` . . . 104
9.5.2 `dput` . . . 104
9.5.3 `dcut` . . . 104
9.6 Автоматизация сопровождения пакетов . . . 104
9.6.1 `devscripts` . . . 104
9.6.2 `reportbug` . . . 104
9.6.3 `autotools-dev` . . . 104
9.6.4 `dpkg-repack` . . . 105

9.6.5 alien . . . 105
9.6.6 elpa-dpkg-dev-el . . . 105
9.6.7 dpkg-depcheck . . . 105
9.6.8 debputy . . . 105
9.7 Инструменты для переноса . . . 106
9.7.1 dpkg-cross . . . 106
9.8 Документация и информацию . . . 106
9.8.1 debian-policy . . . 106
9.8.2 doc-debian . . . 106
9.8.3 developers-reference . . . 106
9.8.4 maint-guide . . . 107
9.8.5 debmake-doc . . . 107
9.8.6 packaging-tutorial . . . 107
9.8.7 how-can-i-help . . . 107
9.8.8 docbook-xml . . . 107
9.8.9 debiandoc-sgml . . . 107
9.8.10 debian-keyring . . . 107
9.8.11 elpa-debian-el . . . 107

Developer's Reference 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

This manual is free software; you may redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2, or (at your option) any later version.

This is distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose. See the GNU General Public License for more details.

A copy of the GNU General Public License is available as /usr/share/common-licenses/GPL-2 in the Debian distribution or on the World Wide Web at the GNU web site. You can also obtain it by writing to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

This is Debian Developer's Reference version `14.18`, released on `2026-11-02`.

If you want to print this reference, you should use the pdf version. This manual is also available in some other languages.

Глава 1

Границы данного документа

The purpose of this document is to provide an overview of the recommended procedures and the available resources for Debian developers and maintainers.

The procedures discussed within include how to become a member (*Applying to Become a Member*); how to create new packages (*Новые пакеты*) and how to upload packages (*Загрузка пакета*); how to handle bug reports (*Работа с ошибками*); how to move, remove, or orphan packages (*Перемещение, удаление, переименование, придание статуса осиротевшего, усыновление и повторное введение пакетов*); how to port packages (*Работа на переносом и перенос пакетов*); and how and when to do interim releases of other maintainers' packages (*Загрузки не-сопровождающим (NMU)*).

Обсуждаемые ресурсы в данном руководстве включают в себя списки рассылки (*Списки рассылки*) и серверы (*Машины Debian*); обсуждается структура архива Debian (*Архив Debian*); объясняются разные серверы, которые принимают загружаемые пакеты (*Загрузка на ftp-master*); также обсуждаются ресурсы, которые могут помочь сопровождающим контролировать качество своих пакетов (*Обзор инструментов Debian для сопровождающего*).

Должно быть ясно, что данное руководство не содержит обсуждения технических деталей пактов Debian и технических деталей их создания. Данное руководство не содержит подробных сведений о стандартах, которым должно отвечать ПО в Debian. Вся эта информация может быть найдена в Руководстве по политике Debian.

Более того, данный документ *не выражает формальной политики*. Он содержит документацию по системе Debian и лучшие практики, принимаемые большинством. Таким образом, это не „нормативный“ документ.

Глава 2

Applying to Become a Member

2.1 Приступаем к работе

So, you've read all the documentation, you've gone through the Debian New Maintainers' Guide (or its successor, Guide for Debian Maintainers), understand what everything in the `hello` example package is for, and you're about to Debianize your favorite piece of software. How do you actually become a Debian developer so that your work can be incorporated into the Project?

Firstly, subscribe to `debian-devel@lists.debian.org` if you haven't already. Send the word `subscribe` in the `Subject` of an email to `debian-devel-REQUEST@lists.debian.org`. In case of problems, contact the list administrator at `listmaster@lists.debian.org`. More information on available mailing lists can be found in *Списки рассылки*. `debian-devel-announce@lists.debian.org` is another list, which is mandatory for anyone who wishes to follow Debian's development.

Вам следует подписаться и ненадолго скрыться (то есть, читать сообщения и ничего не отправлять в список) до начала какой-либо работы над кодом, вам следует писать о ваших намерениях по реализации чего-либо для того, чтобы избежать повторов в работе.

Ещё одним хорошим списком рассылки, на который стоит подписаться, является `debian-mentors@lists.debian.org`. Подробности см. в *Наставники и поручители Debian*. IRC-канал `#debian` также может быть полезен, см. *Каналы IRC*.

Когда вы решите, как вы хотите участвовать в Debian, вам следует связаться с существующими сопровождающими Debian, которые работают над схожими задачами. Это позволит вам научиться у опытных разработчиков. Например, если вы заинтересованы в создании пакетов Debian для существующего ПО, вам следует попытаться найти поручителя. Поручитель будет работать с вами над вашим пакетом и загрузит его в архив Debian, когда он будет доволен проделанной вами работой по созданию пакета. Вы можете найти поручителя отправив сообщение в список рассылки `debian-mentors@lists.debian.org` с описанием вашего пакета и себя самого и с просьбой о поручительстве (для дополнительной информации о поручительстве см. *Поручение пакетов* и https://wiki.debian.org/DebianMentorsFaq). С другой стороны, если вы заинтересованы в переносе Debian на альтернативные архитектуры или ядра, вы можете подписаться на конкретные списки рассылки и спросить там о том, как начать работу. Наконец, если вы заинтересованы в работе над документацией или в контроле качества (QA), вы можете присоединиться к сопровождающим, которые уже работают над этими задачами и отправляют заплаты и улучшения.

One pitfall could be a too-generic local part in your email address: Terms like mail, admin, root, master should be avoided, please see https://www.debian.org/MailingLists/ for details.

2.2 Наставники и поручители Debian

Список рассылки `debian-mentors@lists.debian.org` создан для начинающих сопровождающих, которые ищут помощи в работе над созданием пакетов или в решении связанных с разработкой проблем. Каждому новому разработчику рекомендуется подписаться на этот список рассылки, подробнее см. *Списки рассылки*.

Те, кто предпочитает помощь в режиме «один-на-один» (напр., через частную электронную почту), также должны написать в этот список рассылки и опытные разработчики помогут вам.

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.

Если вы хотите выступить в качестве наставника и/или поручителя, см. дополнительную информацию в *Взаимодействие с будущими разработчиками Debian*.

2.3 Registering as a Debian member

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.

The process of registering as a member is a process of verifying your identity and intentions, and checking your technical skills. As the number of people working on Debian has grown to over 1000 and our systems are used in several very important places, we have to be careful about being compromised. Therefore, we need to verify new members before we can give them accounts on our servers and let them upload packages.

До того как вы фактически зарегистрируетесь, вам следует показать, что вы можете выполнять компетентную работу и быть хорошим участником разработки. Вы можете показать это, отправляя заплаты через систему отслеживания ошибок и работая некоторое время над пакетом, имеющим поручителем существующего разработчика Debian. Кроме того, мы ожидаем, что участники заинтересованы во всём проекте целиком, а не просто в сопровождении своих собственных пакетов. Если мы можете помочь другим сопровождающим, предоставляя дополнительную информацию об ошибке или даже заплату, сделайте это!

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.)

If you do not have an OpenPGP key yet, generate one. Every developer needs an OpenPGP key in order to sign and verify package uploads. You should read the manual for the software you are using, since it has much important information that is critical to its security. Many more security failures are due to human error than to software failure or high-powered spy techniques. See *Сопровождение вашего открытого ключа* for more information on maintaining your public key.

Debian uses the `GNU Privacy Guard` (package `gnupg` version 2 or better) as its baseline standard. You can use some other implementation of OpenPGP as well. Note that OpenPGP is an open standard based on RFC 9580.

Your key length must be greater than 2048 bits (4096 bits is preferred); there is no reason to use a smaller key, and doing so would be much less secure.

Если ваш открытый ключ не размещён на сервере открытых ключей, таком как `subkeys.pgp.net`, прочтите доступную по адресу Шаг 2: Установление личности информацию. Этот документ содержит инструкции о том, как поместить ваш ключ на серверы открытых ключей. Группа новых сопровождающих поместит ваш открытый ключ на серверы, если ключ ещё не был там размещён.

Некоторые страны ограничивают использование ПО для шифрования своими гражданами. Это не должно затруднить вашу деятельность как сопровождающего пакетов Debian, поскольку допускается использование продуктов шифрования для аутентификации, а не в целях зашифровать что-либо. Если вы живёте в стране, в которой запрещено использование шифрования даже для аутентификации, свяжитесь с нами, чтобы мы могли подготовить специальные договорённости.

To apply as a new member, you need an existing Debian Developer to support your application (an `advocate`). After you have contributed to Debian for a while, and you want to apply to become a registered developer, an existing developer with whom you have worked over the past months has to express their belief that you can contribute to Debian successfully.

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.

For more details, please consult New Members Corner at the Debian web site. Make sure that you are familiar with the necessary steps of the New Member process before actually applying. If you are well prepared, you can save a lot of time later on.

Глава 3

Обязанности разработчика Debian

3.1 Обязанности сопровождающего пакетов

As a package maintainer, you're supposed to provide high-quality packages that are well integrated into the system and that adhere to the Debian Policy.

3.1.1 Работа по подготовке следующего `стабильного` выпуска

Providing high-quality packages in `unstable` is not enough; most users will only benefit from your packages when they are released as part of the next `stable` release. You are thus expected to collaborate with the release team to ensure your packages get included.

Более конкретно, вам необходимо отслеживать, мигрировали ли ваши пакеты в `тестируемый` выпуск (см. *Тестируемый выпуск*). Если миграция не произошла после периода тестирования, вам следует выяснить, почему так получилось, и поработать над исправлением проблемы. Это может предполагать исправление пакета (в случае критичных для выпуска ошибок или проблем сборки на некоторых архитектурах), но также это может предполагать обновление (или исправление, или удаление из `тестируемого` выпуска) других пакетов, чтобы помочь в завершении перехода пакетов, в котором ваш пакет застрял из-за своих зависимостей. Команда по выпуску может предоставить вам некоторую информацию о факторах, блокирующих некоторый данный переход пакетов, в случае, если вы сами не можете их определить.

3.1.2 Сопровождение пакетов в `стабильном` выпуске

Большая часть работы сопровождающего пакетов сводится к загрузке обновлённых версий пакетов в `нестабильный` выпуск, но также предполагается отслеживание пакетов в текущем `стабильном` выпуске.

Хотя вносить изменения в `стабильный` выпуск не рекомендуется, они возможны. Когда сообщается о проблеме безопасности, вам следует совместно с командой безопасности предоставить исправленную версию (см. *Работа с ошибками, связанными с безопасностью*). Если сообщается об ошибках с важностью important (или более) в `стабильной` версии вашего пакета, вам следует подумать над предоставлением целевого исправления. Вы можете спросить команду `стабильного` выпуска о том, примут они такое обновление или нет, и затем уже подготовить загрузку в `стабильный` выпуск (см. *Специальный случай: загрузка в стабильный и предыдущий стабильный выпуски*).

3.1.3 Работа с критичными для выпуска ошибками

Обычно вам следует работать с сообщениями об ошибках в ваших пакетах так, как это описано в *Работа с ошибками*. Тем не менее, имеется специальная категория ошибок, на которые следует обратить внимание — это так называемые критичные для выпуска ошибки (RC ошибки). Все такие сообщения об ошибках, имеющие важность `critical`, `grave` или `serious`, делают пакет неподходящим для включения в следующий `стабильный` выпуск. Таким образом, они могут задержать выпуск Debian (когда они затрагивают пакет из `тестируемого` выпуска), либо блокировать миграцию пакетов в `тестируемый` выпуск (когда они лишь затрагивают пакет из `нестабильного` выпуска). В худшем сценарии такие ошибки приведут к удалению пакета. Вот почему эти ошибки следует исправить как можно скорее.

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 *Загрузки не-сопровождающим (NMU)*) because they can block the `testing` migration of many packages.

Отсутствие внимания к критичным для выпуска ошибкам обычно интерпретируется командой контроля качества как знак того, что сопровождающий прекратил свою работу без корректного придания своему пакету статуса осиротевшего пакета. Команда по поиску пропавших (MIA) может подключиться к работе, что может привести к тому, что ваши пакеты получат статус осиротевших пакетов (см. *Работа с неактивными и/или недоступными сопровождающими*).

3.1.4 Координация с разработчиками основной ветки

A big part of your job as Debian maintainer will be to stay in contact with the upstream developers. Debian users will sometimes report bugs that are not specific to Debian to our bug tracking system. These bug reports should be forwarded to the upstream developers so that they can be fixed in a future upstream release. Usually it is best if you can do this, but alternatively, you may ask the bug submitter to do it. Ideally, you also get upstream to *subscribe* for Debian Package Tracker notifications for their software in Debian.

Хотя вашей задачей не является исправление ошибок, которые не касаются непосредственно Debian, вы вполне можете это делать, если имеете необходимые знания и навыки. Когда вы вносите такие исправления, обязательно передайте их и сопровождающим основной ветки разработки. Пользователи и разработчики Debian иногда присылают заплаты, исправляющие ошибки основной ветки разработки — вам следует оценить их и переслать в основную ветку разработки.

In cases where a bug report is forwarded upstream, it may be helpful to remember that the bts-link service can help with synchronizing states between the upstream bug tracker and the Debian one.

Если вам необходимо изменить исходный код из основной ветки разработки для того, чтобы собрать пакет, соответствующий правилам, то вам следует предложить своё исправление разработчикам основной ветки, которые может быть включено ими в основную ветку разработки, чтобы вам не пришлось изменять исходный код последующих выпусков основной ветки. Какие бы изменения вы не производили, попытайтесь не создавать ответвлений от основной ветки.

As most upstreams nowadays use git for version control, in most cases git-buildpackage offers the most convenient way to create and maintain patches in Debian that so they are submit upstream. For details, see git-buildpackage man pages about using `pq` to write and test `debian/patches` as git commits, and having git remote `upstreamvcs` to easily cherry-pick patches to and from upstream git branches.

Если вы обнаружите, что разработчики основной ветки враждебны по отношению к Debian или сообществу Свободного ПО, ещё раз подумайте о том, так ли необходимо включать данное ПО в Debian. Иногда социальная стоимость, которую платит сообщество Debian, не покрывается прибылью, которую может принести какое-либо ПО.

3.2 Административные обязанности

Проект, имеющий размеры, сопоставимые с Debian, полагается на некоторую административную инфраструктуру для того, чтобы всё можно было отследить. Как у члена проекта у вас будут некоторые обязанности, исполнение которых гарантирует, что всё идёт гладко.

3.2.1 Сопровождение вашей связанной с Debian информации

База данных LDAP, содержащая информацию о разработчиках Debian, расположена по адресу https://db.debian.org/. Вам следует ввести информацию о себе в эту базу данных и обновлять её по мере изменения. В первую очередь, убедитесь, что адрес электронной почты, на который пересылается ваша почта с ящика на debian.org, актуален, также проверьте ваш адрес, на который оформлена подписка на рассылку debian-private, если вы подписаны на неё.

Дополнительную информацию о базе данных см. в *База данных разработчиков*.

3.2.2 Сопровождение вашего открытого ключа

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

Вам следует убедиться в том, что ваш ключ не только защищён от кражи, но также и в том, что он не может быть потерян. Создайте и сделайте копию (лучше всего сделать также и бумажную копию) вашего сертификата отзыва ключа; он понадобится в случае потери ключа.

If you add signatures to your public key, or add user identities, you can update the Debian key ring by sending your key to the key server at `keyring.debian.org`. Updates are processed at least once a month by the `debian-keyring` package maintainers.

If you need to add a completely new key or remove an old key, you need to get the new key signed by another developer. If the old key is compromised or invalid, you also have to add the revocation certificate. If there is no real reason for a new key, the Keyring Maintainers might reject the new key. Details can be found at https://keyring.debian.org/replacing_keys.html.

Применимы процедуры разворачивания ключа, обсуждаемые в *Registering as a Debian member*.

You can find a more in-depth discussion of Debian key maintenance in the documentation of the `debian-keyring` package and the https://keyring.debian.org/ site.

3.2.3 Голосование

Даже несмотря на то, что Проект Debian в действительности не представляет собой демократию, мы используем демократический процесс для выбора лидеров Проекта и утверждения общих решений. Эти процедуры определяются Конституцией Проекта Debian.

Помимо ежегодных выборов лидера Проекта, голосования не проводятся регулярно, и к ним не относятся легкомысленно. Каждое предложение сначала обсуждается в списке рассылки `debian-vote@lists.debian.org`, требуется некоторое одобрение любого предложения до того, как секретарь Проекта инициирует процедуру голосования.

You don't have to track the pre-vote discussions, as the secretary will issue several calls for votes on `debian-devel-announce@lists.debian.org` (and all developers are expected to be subscribed to that list). Democracy doesn't work well if people don't take part in the vote, which is why we encourage all developers to vote. Voting is conducted via OpenPGP-signed/encrypted email messages.

Список всех предложений (прошлых и текущих) доступен на странице Информации о голосованиях Debian, там же доступна информация о том, как выдвинуть, поддержать и проголосовать по поводу предложения.

3.2.4 Decision making

Besides the issues covered in *Голосование*, Debian is closer to a do-ocracy, than a democracy. Generally speaking, this means that decisions are taken by the people who do the work, which in most cases is package maintainers and DPL delegates. Public discussion:

- while encouraged and may be appreciated as input to take a decision,
- is not required to take a decision,
- nor can overrule a decision (the latter is possible via a General Resolution or by escalating to the Debian Technical Committee).

In the spirit of do-ocracy, members of the Debian community are expected to contribute in concrete and constructive ways, rather than demand that others do any kind of work for them.

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 Вежливый уход в отпуск

Разработчики могут отсутствовать, это нормально, это может быть запланированный отпуск, либо они могут быть загружены другой работой. Важно, чтобы остальные разработчики знали о том, что вы находитесь в отпуске, чтобы они могли сделать то, что нужно, если возникнут проблемы в ваших пакетах, либо от вас требуется выполнение каких-либо других обязанностей в Проекте.

Обычно это означает, что другие разработчики могут осуществлять загрузки, не будучи сопровождающими (см. *Загрузки не-сопровождающим (NMU)*), в случае если во время вашего отсутствия возникнет какая-либо большая проблема (критичная для выпуска ошибка, обновление безопасности и т. д.). Иногда ничего критичного не происходит, но всё равно лучше предупредить остальных о том, что вы будете недоступны.

Для того, чтобы проинформировать других разработчиков, вам следует сделать две вещи. Во-первых, отправьте сообщение по адресу `debian-private@lists.debian.org` с [VAC] в начале темы сообщения[1], в сообщении укажите промежуток времени, когда вы будете находиться в отпуске. Вы также можете указать какие-либо специальные инструкции по поводу того, что нужно предпринять в случае, если возникнет проблема.

Далее, следует отметить себя как находящегося в отпуске в *База данных разработчиков* (эта информация доступна только разработчикам Debian). Не забудьте удалить флаг статуса «в отпуске» по своему возвращению!

Ideally, you should sign up at the OpenPGP coordination pages when booking a holiday and check if anyone there is looking for signing. This is especially important when people go to exotic places where we don't have any developers yet but where there are people who are interested in applying.

3.2.6 Уход в отставку

Если вы решили покинуть Проект Debian, вам следует убедиться, что вы выполнили следующие шаги:

- Orphan all your packages, as described in *Придание статуса осиротевшего пакета*.
- Remove yourself from uploaders for co- or team-maintained packages.
- Если вы получали сообщения через алиас @debian.org (напр. press@debian.org) и хотите удалить свой адрес из рассылки, откройте билет RT для системных администраторов Debian. Просто отправьте сообщение на адрес `admin@rt.debian.org` со словами "Debian RT" где-нибудь в теме сообщения, в сообщении укажите, от каких алиасов вы более не хотите получать сообщения.
- Please remember to also retire from teams, e.g. remove yourself from team wiki pages or salsa groups.

[1] Это нужно для того, чтобы те, кто не хочет читать сообщения об отпусках, могли отфильтровать такие сообщения.

- 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`

Важно, чтобы весь приведённый выше процесс был соблюдён, поскольку поиск неактивных разработчиком и пометка их пакетов как осиротевших занимает значительное количество времени и требует усилий.

3.2.7 Возвращение после ухода

A retired developer's account is marked as "emeritus" when the process in *Уход в отставку* is followed, and "removed" otherwise. Retired developers with an "emeritus" account can get their account reactivated as follows:

- Log on onto nm.debian.org using your salsa credentials.
- Follow the Return from Emeritus Wizard.
- Пройдите через укороченную процедуру получения статуса нового члена (это необходимо для того, чтобы убедиться, что возвращающийся разработчик всё ещё помнит важные части P&P и T&S).

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

Retired developers with a "removed" account need to go through full NM again.

Глава 4

Resources for Debian Members

In this chapter you will find a very brief roadmap of the Debian mailing lists, the Debian machines which may be available to you as a member, and all the other resources that are available to help you in your work.

4.1 Списки рассылки

Большая часть общения между разработчиками Debian (а также пользователями) происходит в списках рассылки, которые расположены по адресу `lists.debian.org`. Чтобы узнать о том, как подписаться или отписаться, как отправлять сообщения и как не отправлять их, где искать старые сообщения и как осуществлять поиск по ним, как связаться с сопровождающими данного списка рассылки и как просмотреть другую информацию о данном списке рассылки, прочтите https://www.debian.org/MailingLists/. В данном разделе приводятся лишь некоторые аспекты списков рассылки, которые особенно интересны разработчикам.

4.1.1 Базовые правила

При ответе на сообщения в списке рассылки не отправляйте копию сообщения (`CC`) автору сообщения, на которое вы отвечаете, если он об этом не попросил явным образом. Всякий, кто отправляет сообщения в список рассылки, должен быть подписан на него, чтобы видеть ответы.

Отправление нескольких одинаковых сообщений в разные списки рассылки не желательно. Как это обычно делается при работе в сети, сокращайте цитаты из сообщений, на которые вы отвечаете. В общем придерживайтесь обычных договорённостей по отправке сообщений.

Для получения дополнительной информации ознакомьтесь с нормами поведения. Также стоит прочитать Руководство по взаимодействию в сообществе Debian.

4.1.2 Базовые списки рассылки

Базовые списки рассылки Debian, которыми следует пользоваться разработчикам:

- `debian-devel-announce@lists.debian.org` используется для анонсирования важной для разработчиков информации. Ожидается, что все разработчики подписаны на этот список рассылки.
- `debian-devel@lists.debian.org` используется для обсуждения различных вопросов разработки, связанных с техническими проблемами.

- `debian-policy@lists.debian.org` используется для обсуждения Политики Debian, а также для голосования по её изменению.
- `debian-project@lists.debian.org` используется для обсуждения различных не технических проблем, связанных с Проектом.

Существуют и другие списки рассылки для обсуждения специальных вопросов; см. список по адресу https://lists.debian.org/.

4.1.3 Специальные списки рассылки

`debian-private@lists.debian.org` представляет собой специальный список рассылки для частных дискуссий между разработчиками Debian. Предполагается, что он будет использоваться для сообщений, которые по какой-либо причине не должны быть доступны широкой публике. Как таковой это небольшой список рассылки, пользователям рекомендуется не использовать `debian-private@lists.debian.org`, если только это не является действительно необходимым. Более того, *не* пересылайте сообщения из этого списка кому бы то ни было. Архивы данного списка не доступны в сети по очевидным причинам, но вы можете просмотреть их, используя вашу учётную запись для командной оболочки на `master.debian.org`, архив находится в каталоге `~debian/archive/debian-private/`.

`debian-email@lists.debian.org` представляет собой специальный список рассылки, используемый для сбора связанной с Debian корреспонденции, такой как взаимодействие с авторами основной ветки разработки по поводу лицензий, ошибок и т. д., либо для обсуждения Проекта с другими пользователями и разработчиками, если сохранение этого обсуждения может оказаться полезным.

4.1.4 Запрос новых списков рассылки, связанных с разработкой

Before requesting a mailing list that relates to the development of a package (or a small group of related packages), please consider if using an alias (via a .forward-aliasname file on master.debian.org, which translates into a reasonably nice *you-aliasname@debian.org* address) is more appropriate.

Если вы решили, что обычный список рассылки на lists.debian.org — это то, что вы хотите, то заполните запрос, следуя данному руководству.

4.2 Каналы IRC

Несколько каналов IRC специально посвящены разработке Debian. В основном они размещены в сети сообщества открытых и свободных технологий (OFTC). Запись DNS `irc.debian.org` является псевдонимом для `irc.oftc.net`.

Основным общим каналом для Debian является `#debian`. Это большой канал общего назначения, где пользователи могут найти свежие новости о Проекте, этот канал обслуживается роботами. `#debian` предназначен для говорящих на английском языке; также имеются `#debian.de`, `#debian-fr`, `#debian-br` и другие каналы со сходными названиями для тех, кто говорит на других языках.

Основным каналом по вопросам разработки Debian является `#debian-devel`. Это довольно активный канал; обычно на нём присутствует не менее 150 человек в любое время дня. Это канал для тех, кто хочет работать над Debian, это не канал поддержки (это этого используется канал `#debian`). Тем не менее, этот канал открыт для всех, кто хочет затаиться (и научиться). Его тематика состоит в обсуждении любой информации, которая интересна разработчикам.

Поскольку `#debian-devel` является открытым каналом, вам не следует обсуждать в нём то, что обсуждается в `debian-private@lists.debian.org`. Для этого цели имеется другой канал, который имеет имя `#debian-private`, он защищён ключом. Ключ доступен по адресу `master.debian.org:~debian/misc/irc-password`.

There are other additional channels dedicated to specific subjects. `#debian-bugs` is used for coordinating bug squashing parties. `#debian-boot` is used to coordinate the work on the debian-installer. `#debian-doc` is occasionally used to talk about documentation, like the document you are reading. Other channels

are dedicated to an architecture or a set of packages: `#debian-kde`, `#debian-dpkg`, `#debian-perl`, `#debian-python`...

Существуют также некоторые каналы для разработчиков, родным языком которых не является английский, например `#debian-devel-fr` для людей, говорящих на французском языке и заинтересованных в разработке Debian.

Channels dedicated to Debian also exist on other IRC networks.

4.3 Документация

This document contains a lot of information which is useful to Debian developers, but it cannot contain everything. Most of the other interesting documents are linked from The Developers' Corner. Take the time to browse all the links; you will learn many more things.

4.4 Машины Debian

У Debian имеется несколько компьютеров, которые работают в качестве серверов, большинство из них обслуживают критически важные функции в Проекте Debian. Большая часть машин используется для переноса пакетов, все эти машины имеют постоянное подключение к сети Интернет.

Некоторые машины доступны для использования отдельными разработчиками до тех пор, пока эти разработчики соблюдают правила, приведённые в Политике использования машин Debian.

Вообще говоря, вы можете использовать эти машины для связанных с Debian целей, если это кажется вам подходящим. Пожалуйста, будьте добры с системными администраторами, не используйте большое количество места на диске, пропускной способности сети, либо процессорного времени без предварительного разрешения системных администраторов. Обычно эти машины обслуживаются добровольцами.

Позаботьтесь о защите ваших паролей Debian и ключей SSH, установленных на машинах Debian. Избегайте входа или методов загрузки, которые пересылают пароли по сети Интернет в открытом виде, такие как Telnet, FTP, POP и т. д.

Пожалуйста, не размещайте какой-либо материал, не связанный с Debian, на серверах Debian, если у вас нет на это соответствующего разрешения.

Текущий список машин Debian доступен по адресу: https://db.debian.org/machines.cgi. Эта веб-страница содержит имена машин, контактную информацию, информацию о том, кто может входить на данную машину, ключи SSH и т. д.

Если у вас имеются проблемы с работой сервера Debian, и вы считаете, что системные операторы должны быть извещены об этой проблеме, вы можете обратиться к списку открытых проблем в очереди DSA в нашей системе отслеживания билетов https://rt.debian.org/ (вы можете войти под пользователем «debian», пароль же доступен по адресу: `master.debian.org:~debian/misc/rt-password`). Чтобы сообщить о новой проблеме, вам нужно лишь отправить сообщение по адресу `admin@rt.debian.org` и убедиться, что в теме вы указали «Debian RT». Чтобы связаться с командой DSA по электронной почте, для частной или привилегированной информации, которая не должна быть публичной, используйте адрес `dsa@debian.org`, а для всего остального адрес `debian-admin@lists.debian.org`. Команда DSA также может быть найдена на IRC-канале `#debian-admin` в сети OFTC.

Если у вас имеются проблемы с конкретной службой, которые не связаны с администрированием системы (такие как пакеты, которые следует удалить из архива, предложения для веб-сайта и т. д.), вам следует отправить сообщение об ошибке в псевдопакете „pseudo-package“. Информацию о том, как отправлять такие сообщения об ошибках, см. *Отправка отчётов об ошибках*.

Некоторые корневые серверы ограничены, но информация с них зеркалируется на других серверах.

4.4.1 Сервер bugs

`bugs.debian.org` является каноническим размещением системы отслеживания ошибок.

Если вы планируете проводить какой-либо статистической анализ, либо обработку сообщений об ошибках Debian, то это можно сделать здесь. Тем не менее, пожалуйста, опишите ваши планы в сообщений в `debian-devel@lists.debian.org` до реализации чего бы то ни было, чтобы уменьшить объём ненужной работы или время на обработку запросов.

4.4.2 Сервер ftp-master

The `ftp-master.debian.org` server holds the canonical copy of the Debian archive. Generally, packages uploaded to ftp.upload.debian.org end up on this server; see *Загрузка пакета*.

Доступ у нему ограничен; зеркало доступно по адресу `mirror.ftp-master.debian.org`.

О проблемах с архивом Debian FTP следует сообщать как об ошибках в псевдопакете `ftp.debian.org`, либо по электронной почте на адрес `ftpmaster@debian.org`, также см. описание процедур в *Перемещение, удаление, переименование, придание статуса осиротевшего, усыновление и повторное введение пакетов*.

4.4.3 Сервер www-master

Основной веб-сервер расположен по адресу `www-master.debian.org`. Он содержит официальные веб-страницы и представляет собой лицо Debian для большинства новичков.

If you find a problem with the Debian web server, you should generally submit a bug against the pseudo-package `www.debian.org`. Remember to check whether or not someone else has already reported the problem to the Bug Tracking System.

4.4.4 Веб-сервер people

`people.debian.org` представляет собой сервер, используемый для размещения собственных веб-страниц разработчиков обо всём, что связано с Debian.

Если у вас имеется какая-то касающаяся Debian информация, которую вы хотели бы разместить в веб, вы можете сделать это, поместив ваш материал в каталог `public_html` своего домашнего каталога на `people.debian.org`. Материал будет доступен по URL: `https://people.debian.org/~`*ваш-идентификатор-пользователя*`/`.

Вам следует использовать именно это место потому, что мы создаём его резервную копию, на других узлах резервные копии не создаются.

Обычно единственная причина использовать другой узел заключается в том, что вы хотите опубликовать какие-то материалы, которые подпадают под ограничения экспорта с территории США, в это случае вы можете использовать один из тех серверов, которые размещены за пределами США.

Если у вас имеются какие-либо вопросы, отправьте сообщения по адресу `debian-devel@lists.debian.org`.

4.4.5 salsa.debian.org: Git repositories and collaborative development platform

If you want to use a git repository for any of your Debian work, you can use Debian's GitLab instance called Salsa for that purpose. Gitlab provides also the possibility to have merge requests, wiki pages, bug trackers among many other services as well as a fine-grained tuning of access permission, to help working on projects collaboratively.

For more information, please see the documentation at https://wiki.debian.org/Salsa/Doc.

Any Debian package hosted on Salsa has also access to the Salsa CI . The Salsa CI pipeline mimics the tests that are run after each upload to Debian, but instead of having to wait for results or risk the health of the Debian repositories, Salsa CI provides you with instant feedback about any problems the changes you made may have created or solved.

4.4.6 GitHub.com: Submitting pull requests to upstream repositories

If some upstream repository is hosted on GitHub.com, you can use the Debian organization to create repository forks and submit changed branches with pull requests to upstream maintainers.

The organization is open to all Debian Members. To request membership, open an issue in the Debian/.github meta repository.

4.5 База данных разработчиков

The Developers Database, at https://db.debian.org/, is an LDAP directory for managing Debian developer attributes. You can use this resource to search the list of Debian developers. Part of this information is also available through the finger service on Debian servers; try `finger yourlogin@db.debian.org` to see what it reports.

Разработчики могут входить в базу данных для изменения различной информации о самих себе, как то:

- forwarding address for your debian.org email as well as spam handling. See https://db.debian.org/forward.html for a description of all the options.
- подписка на debian-private
- нахождение в отпуске
- персональная информация, например, адрес, страна, широта и долгота места, где вы проживаете, и которые будут использованы на мировой карте разработчиков Debian, номера телефонов и факсов, псевдоним в IRC и адрес веб-страницы
- пароль и предпочтительная оболочка на машинах Проекта Debian

Естественно, большая часть информации не доступна публично. Для получения дополнительной информации, прочтите документацию, которая может быть найдена по адресу https://db.debian.org/doc-general.html.

Разработчики могут добавлять свои SSH-ключи, которые будут использовать для авторизации на официальных машинах Debian, и даже добавлять новые записи DNS вида *.debian.net. Эти возможности описаны по адресу https://db.debian.org/doc-mail.html.

4.6 Архив Debian

Дистрибутив Debian состоит из огромного числа пакетов (в настоящее время около 40000 пакетов с исходным кодом) и нескольких дополнительных файлов (таких как документация и образы установочных дисков).

Ниже приведён пример дерева каталогов полного архива Debian:

```
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/
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
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/

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/
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
pool/non-free-firmware/f/firmware-nonfree/
     ...
```

Как вы можете видеть, каталог верхнего уровня содержит два каталога, `dists/` и `pool/`. Последний представляет собой "пул", в котором фактически находятся пакеты, и который обрабатывается базой данных для сопровождения архива и сопутствующими программами. Первый же содержит выпуски, `стабильный`, `тестируемый` и `нестабильный`. Файлы `Packages` и `Sources` в подкаталогах выпусков могут указывать на файлы в каталоге `pool/`. Деревья каталогов каждого выпуска организованы одинаковым образом. То, что мы описываем ниже для `стабильного` выпуска применимо и для `нестабильного`, и для `тестируемого` выпусков.

`dists/stable` contains four directories, namely `main`, `contrib`, `non-free` and `non-free-firmware`.

В каждом из них имеется каталог для пакетов с исходным кодом (`source`) и каталог для каждой поддерживаемой архитектуры (`binary-i386`, `binary-amd64` и т. д.).

Каталог `main` содержит дополнительные каталоги, в которых хранятся образы дисков и некоторые важные части документации, требующиеся для установки дистрибутива Debian на конкретную архитектуру (`disks-i386`, `disks-amd64` и т. д.).

4.6.1 Разделы

Раздел `main` архива Debian — то, что является **официальным дистрибутивом Debian**. Раздел `main` является официальным потому, что он полностью соответствует всем нашим рекомендациям. Другие два раздела не соответствуют им в той или иной степени; как таковые они **не** являются официальной частью Debian.

Каждый пакет из основного раздела должен полностью соответствовать Критериям Debian по определению Свободного ПО (DFSG), а также всем другим правилам, описываемым в Руководстве по политике Debian. DFSG представляет собой наше определение понятия "свободное ПО." Для получения дополнительной информации ознакомьтесь с Руководством по политике Debian.

Пакеты из раздела `contrib` должны соответствовать DFSG, но могут не отвечать другим требованиями. Например, они могут зависеть от несвободных пакетов.

Packages which do not conform to the DFSG are placed in the `non-free` or `non-free-firmware` sections. These packages are not considered as part of the Debian distribution, though we enable their use, and we provide infrastructure (such as our bug-tracking system and mailing lists) for these non-free software packages.

The Debian Policy Manual contains a more exact definition of the four sections. The above discussion is just an introduction.

The separation of the four sections at the top-level of the archive is important for all people who want to distribute Debian, either via FTP servers on the Internet or on CD-ROMs: by distributing only the `main` and `contrib` sections, one can avoid any legal risks. Some packages in the `non-free` section do not allow commercial distribution, for example.

С другой стороны, производитель CD-ROM легко может проверить лицензию отдельно взятого пакета в разделе `non-free` и добавить столько пакетов из этого раздела на CD-ROM, сколько это разрешено их лицензиями. (Поскольку это отличается от производителя к производителю, данная работа не может быть проделана разработчиками Debian.)

Note that the term section is also used to refer to categories which simplify the organization and browsing of available packages: `admin`, `net`, `utils`, etc. Once upon a time, these sections (subsections, rather) existed in the form of subdirectories within the Debian archive. Nowadays, these exist only in the Section header fields of packages.

4.6.2 Архитектуры

В начале ядро Linux было доступно только для платформ Intel i386 (или выше), поэтому и Debian был доступен только для этой платформы. Но когда Linux стал всё более и более популярным, ядро было перенесено на другие архитектуры, и Debian начал их поддерживать. И если этой поддержки оборудования было недостаточно, Debian решал собрать некоторые переносы на основе ядер Unix, например `hurd` и `kfreebsd`.

Debian GNU/Linux 1.3 was only available as `i386`. Debian 2.0 shipped for `i386` and `m68k` architectures. Debian 2.1 shipped for the `i386`, `m68k`, `alpha`, and `sparc` architectures. Since then Debian has grown hugely. Debian 9 supports a total of ten Linux architectures (`amd64`, `arm64`, `armel`, `armhf`, `i386`, `mips`, `mips64el`, `mipsel`, `ppc64el`, and `s390x`) and two kFreeBSD architectures (`kfreebsd-i386` and `kfreebsd-amd64`).

Информация для разработчиков и пользователей о конкретных переносах доступна на веб-страницах переносов Debian.

4.6.3 Пакеты

Имеется два типа пакетов Debian, а именно пакеты `с исходным кодом` и `двоичные` пакеты.

В зависимости от формата пакета с исходным кодом, он состоит из одного или нескольких файлов в дополнение к обязательному файлу `.dsc`:

- формат "1.0" содержит либо файл `.tar.gz`, либо файл `.orig.tar.gz` и файл `.diff.gz`;
- формат "3.0 (quilt)" содержит обязательный tar-архив из основной ветки разработки `.orig.tar.{gz,bz2,xz}`, множество необязательных дополнительных tar-архивов из основной ветки разработки `.orig-`*компонент*`.tar.{gz,bz2,xz}` и обязательный tar-архив debian doc `debian.tar.{gz,bz2,xz}`;
- формат "3.0 (native)" содержит только один tar-архив `.tar.{gz,bz2,xz}`.

If a package is developed specially for Debian and is not distributed outside of Debian, there is just one `.tar.{gz,bz2,xz}` file, which contains the sources of the program; it's called a "native" source package. If a package is distributed elsewhere too, the `.orig.tar.{gz,bz2,xz}` file stores the so-called `upstream source code`, that is the source code that's distributed by the `upstream maintainer` (often the author of the software). In this case, the `.diff.gz` or the `debian.tar.{gz,bz2,xz}` contains the changes made by the Debian maintainer.

Файл `.dsc` содержит список всех файлов в пакете с исходным кодом, а также их контрольные суммы (`md5sums`, `sha1sums`, `sha256sums`) и некоторую дополнительную информацию о пакете (сопровождающий, версия и т. д.).

4.6.4 Выпуски

The directory system described in the previous chapter is itself contained within `distribution directories`. Each distribution is actually contained in the `pool` directory in the top level of the Debian archive itself.

To summarize, the Debian archive has a root directory within a mirror site. For instance, at the mirror site `ftp.us.debian.org` the Debian archive itself is contained in /debian, which is a common location (another is `/pub/debian`).

Выпуск включает в себя исходный код Debian и двоичные пакеты, а также соответствующие индексные файлы `Sources` и `Packages`, содержащие заголовочную информацию всех пакетов. Первый файл хранится в каталоге `pool/`, а последний — в каталоге `dists/` архива (для обратной совместимости).

4.6.4.1 Стабильный, тестируемые и нестабильный выпуски

Всегда имеются выпуски, называемые `стабильным` выпуск (он расположен в `dists/stable`), `тестируемый` выпуск (он расположен в `dists/testing`), а также `нестабильный` выпуск (он расположен в `dists/unstable`). Эта иерархия отражает процесс разработки Проекта Debian.

Активная разработка выполняется в `нестабильном` выпуске (вот почему этот выпуск иногда называется `разрабатываемым выпуском`). Каждый разработчик Debian может загружать свои пакеты в этот выпуск в любое время. Таким образом, содержимое этого выпуска меняется день ото дня. Поскольку для того, чтобы убедится, что этот выпуск работает нормально, иногда он в буквальном смысле бывает нестабильным.

Тестируемый выпуск создаётся автоматически путём принятия пакетов из `нестабильного` выпуска, если они удовлетворяют определённым критериям. Эти критерии должны гарантировать хорошее качестве пакетов в `тестируемом` выпуске. Обновление `тестируемого` выпуска запускается дважды в день, сразу после установки новых пакетов. См. *Тестируемый выпуск*.

После периода разработки, когда управляющий выпуском решает, что время подходящее, `тестируемый` выпуск замораживается, что означает, что правила, управляющие тем, как пакеты переходят из `нестабильного` выпуска в `тестируемый` ужесточаются. Удаляются пакеты с большим количеством ошибок. Запрещается изменять что-либо в `тестируемом` выпуске за исключением исправления ошибок. Через некоторое время, в зависимости от хода всего процесса, `тестируемый` выпуск переходит на вторую стадию заморозки. Подробности работы с тестируемым выпуском публикуются выпускающей командой в списке рассылки debian-devel-announce. Когда открытые проблемы будут решены так, что выпускающая команда будет удовлетворена, осуществляется выпуск дистрибутива. Это предполагает, что `тестируемый` выпуск переименовывается в `стабильный` выпуск, создаётся его копия для нового `тестируемого` выпуска, а предыдущий `стабильный` переименовывается в `предыдущий стабильный` выпуск и остаётся им до тех пор пока наконец не будет архивирован. Во время архивирования его содержимое перемещается на `archive.debian.org`.

Этот цикл разработки основывается на допущении, что `нестабильный` выпуск становится `стабильным` после периода `тестирования`. Даже если выпуск считается стабильным, в нём всё равно присутствуют ошибки — вот почему стабильный выпуск продолжает обновляться. Тем не менее, эти обновления довольно тщательно тестируются до их включения в архив и добавляются по одному для того, чтобы снизить риск добавления новых ошибок. Вы можете найти предлагаемые дополнения к `стабильному` выпуску в каталоге `proposed-updates`. Те пакеты из `proposed-updates`, которые прошли проверку, периодически перемещаются в составе группы других пакетов в стабильный выпуск, а номер редакции стабильного выпуска увеличивается (напр., '6.0' становится '6.0.1', '5.0.7' становится '5.0.8' и т. д.). За подробностями обратитесь к разделу *Специальный случай: загрузка в стабильный и предыдущий стабильный выпуски*.

Note that development in `unstable` during the freeze should not be continued as usual, as packages are still build in `unstable`, before they migrate to `testing`, thus `unstable` should only contain packages meant for `testing`. Thus only upload to `unstable` during freezes, if you are planning to request an unblock (or if the package is not in `testing`).

If you want to develop new stuff for after the freeze, upload to `experimental` instead.

4.6.4.2 Дополнительная информация о тестируемом выпуске

Обычно пакеты устанавливаются в `тестируемый` выпуск после того, как они пройдут некоторое тестирование в `нестабильном` выпуске.

Для получения дополнительных сведений обратитесь к *Тестируемый выпуск*.

4.6.4.3 Экспериментальный выпуск

`Экспериментальный` выпуск является специальным выпуском. Это не полный выпуск в том же смысле как являются полными `стабильный`, `тестируемый` и `нестабильный` выпуски. Наоборот, это лишь временное место для экспериментального ПО, которое вполне может сломать ваше систему, либо для ПО, которое пока ещё недостаточно стабильно для того, чтобы помещать его в `нестабильный` выпуск (но всё равно имеются причины для создания такого пакета). Ожидается, что пользователи, которые скачивают и устанавливают пакеты из `экспериментального` выпуска, предупреждены

о возможных проблемах. Короче, для экспериментального выпуска мы не даём никаких гарантий, вся ответственность исключительно на вас.

Вот строки sources.list 5 для экспериментального выпуска:

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

Если имеется возможность того, что ПО может нанести непоправимый вред системе, лучше всего поместить его в экспериментальный выпуск. Например, поддержка экспериментальной файловой системы с сжатием должна, вероятно, войти в экспериментальный выпуск.

Если имеется новая версия пакета из основной ветки разработки, которая добавляет новые возможности, но приводит к поломке старых возможностей, она либо не должна быть загружена, либо должна быть загружена в экспериментальный выпуск. Новая бета версия некоторого ПО, использующая совершенно другую настройку, может войти в экспериментальный выпуск решению сопровождающего. Если вы работаете над несовместимой или сложной ситуацией по обновлению пакета, вы также можете использовать экспериментальный выпуск для тестирования, так чтобы те, кто будут тестировать ваш пакет, могли раньше получить к нему доступ.

Некоторое экспериментальное ПО может войти и в нестабильный выпуск, если в описание вы добавите предупреждение, но это не рекомендуется, поскольку пакеты предполагается, что пакеты из нестабильного выпуска будут перемещены в тестируемый выпуск, а затем в стабильный. Вам не следует бояться использовать экспериментальный выпуск, поскольку он не доставляет проблем сопровождающим ftp, экспериментальные пакеты периодически удаляются как только вы загрузите в нестабильный выпуск пакет, имеющий более высокий номер версии.

Новое ПО, которое скорее всего не повредит систему, может быть сразу добавлено в нестабильный выпуск.

Альтернативой экспериментальному выпуску является ваше личное пространство на people.debian.org.

4.6.5 Кодовые имена выпусков

Every released Debian distribution has a code name: Debian 11 is called bullseye; Debian 12, bookworm; Debian 13, trixie; the next release, Debian 14, will be called forky and Debian 15 will be called duke. There is also a *pseudo-distribution*, called sid, which is the current unstable distribution; since packages are moved from unstable to testing as they approach stability, sid itself is never released. As well as the usual contents of a Debian distribution, sid contains packages for architectures which are not yet officially supported or released by Debian. These architectures are planned to be integrated into the mainstream distribution at some future date. The codenames and versions for older releases are listed on the website.

Поскольку Debian следует открытой модели разработки (т. е., всякий может участвовать и следить за разработкой), даже нестабильный и тестируемый выпуски распространяются в Интернет через сеть FTP и HTTP серверов Debian. Таким образом, если мы назвали каталог, содержащий версию, рассматриваемую в качестве кандидата на выпуск, testing, то мы переименуем его в stable, когда эта версия будет выпущена, что приведёт к тому, что все FTP зеркала заново загрузят весь дистрибутив (который довольно велик).

С другой стороны, если мы с самого начала назовём каталоги выпусков Debian-x.y, люди будут полагать, что доступен выпуск Debian версии x.y. (Такое было в прошлом, когда производитель CD-ROM собрал CD-ROM Debian 1.0 на основе разрабатываемой версии pre-1.0. Вот почему первым официальным выпуском Debian был 1.1, а не 1.0.)

Таким образом, имена каталогов для выпусков в архиве соответствуют кодовым именам выпусков, а не статусу выпуска (напр., trixie). Эти имена остаются одними и теми же в период разработки в после выпуска; символические ссылки, которые легко могут быть изменены, обозначают текущий стабильный выпуск. Вот почему фактические каталоги используют кодовые имена, а символические ссылки для стабильного, тестируемого и нестабильного выпусков указывают на соответствующие каталоги выпусков.

4.7 Зеркала Debian

Разные архивы для скачивания и веб-сайт имеют несколько зеркал для того, чтобы освободить наши каноничные серверы от тяжёлой нагрузки. Фактически, некоторые каноничные серверы не доступны публично — вместо этого первый ряд зеркал занимается балансировкой нагрузки. Так, пользователи всегда получают доступ к зеркалу и привыкают использовать их, что позволяет Debian лучше разделять пропускную способность между несколькими серверами и сетями и вообще позволяет пользователям избежать обращения к основному серверу. Заметьте, что первый ряд серверов является наиболее актуальными, поскольку они обновляются по запросу с внутренних сайтов (мы называем это проталкивающим зеркалированием).

Все информация о зеркалах Debian, включая список доступных публично FTP/HTTP серверов, может быть найдена на https://www.debian.org/mirror/. Эта полезная страница содержит информацию и инструменты, которые могут быть вам полезны, если вы заинтересованы в настройке собственного зеркала, как для внутренних нужд, так и с публичным доступом.

Note that mirrors are generally run by third parties who are interested in helping Debian. As such, developers generally do not have accounts on these machines.

4.8 Система входящих пакетов

Система входящих пакетов ответственна за сбор обновлённых пакетов и их установку в архив Debian. Она состоит из набора каталогов и сценариев, которые установлены на `ftp-master.debian.org`.

Пакеты загружаются сопровождающими в каталог с названием `UploadQueue`. Этот каталог сканируется каждые несколько минут службой, названой `queued`, выполняются файлы `*.command`, а оставшиеся и корректно подписанные файлы `*.changes` перемещаются вместе с соответствующими их файлами в каталог `unchecked`. Этот каталог невидим для большинства разработчиков, поскольку доступ у главному ftp ограничен; он сканируется каждые 15 минут сценарием `dak process-upload`, который проверяет целостность загруженных пакетов и их криптографические подписи. Если пакет считается готовым к установке, он перемещается в каталог `done`. Если это первая загрузка данного пакета (либо в ней содержаться новые двоичные пакеты), пакет перемещается в каталог `new`, где он должен ожидать подтверждения от сопровождающих ftp. Если пакет содержит файлы, которые должны быть установлены вручную, он перемещается в каталог `byhand`, где он ожидает ручной установки, выполняемой сопровождающими ftp. В противном случае, если была обнаружена какая-либо ошибка, пакет получает отказ и перемещается в каталог `reject`.

Когда пакет будет принят, система отправляет сообщение с подтверждением этого сопровождающему и закрывает все ошибки, которые были отмечены как исправленные в данной загрузке, тогда ПО для автоматической сборки может начать их повторную компиляцию. Теперь пакет доступен публично в https://incoming.debian.org/, она остаётся там до фактической установки в архив Debian. Это происходит четыре раза в день (и также по историческим причинам называется `dinstall run`); затем пакет удаляется из входящих и устанавливается в пул вместе со всеми другими пакетами. Когда все другие обновления (создающие, например, индексные файлы `Packages` и `Sources`) будут произведены, вызывается специальный сценарий, которые просит все первичные зеркала запустить обновление.

The archive maintenance software will also send the OpenPGP signed `.changes` file that you uploaded to the appropriate mailing lists. If a package is released with the `Distribution` set to `stable`, the announcement is sent to `debian-changes@lists.debian.org`. If a package is released with `Distribution` set to `unstable` or `experimental`, the announcement will be posted to `debian-devel-changes@lists.debian.org` or `debian-experimental-changes@lists.debian.org` instead.

Хотя доступ к главному ftp ограничен, копия установки доступна всем разработчикам по адресу `mirror.ftp-master.debian.org`.

4.9 Информация о пакете

4.9.1 В веб

Каждый пакет имеет несколько выделенных для него веб-страниц. `https://packages.debian.org/`*имя-пакета* отображает каждую версию пакета, доступную в различных выпусках. Каждая версия представляет собой ссылку на страницу, предоставляющую информацию, включая описание пакета, зависимости и ссылки для скачивания пакета.

Система отслеживания ошибок отслеживает ошибки каждого пакета. Вы можете просмотреть ошибки любого данного пакета по адресу `https://bugs.debian.org/`*имя-пакета*.

4.9.2 Утилита `dak ls`

`dak ls` является частью набора инструментов dak, она выводит список доступных версий пакета для всех известных выпусков и архитектур. Инструмент `dak` доступен на `ftp-master.debian.org`, а также на зеркале `mirror.ftp-master.debian.org`. Он используется единственный аргумент, соответствующий имени пакета. Пример объяснит это лучше:

```
$ 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
```

В этом примере вы можете видеть, что версия в `нестабильном` выпуске отличается от версии в `тестируемом` выпуске, и что была сделана binNMU этого пакета для всех архитектур. Каждая версия пакета была заново скомпилирована на всех архитектурах.

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 Система отслеживания пакетов Debian

The Debian Package Tracker is an email and web-based tool to track the activity of a source package. You can get the same emails that the package maintainer gets, simply by *subscribing* to the package in the Debian Package Tracker.

Система отслеживания пакетов имеет веб-интерфейс по адресу https://tracker.debian.org/, в нём собирается множество информации о каждом пакете с исходным кодом. Там имеется множество полезных ссылок (система отслеживания ошибок, статистика QA, контактная информация, статус перевода DDTP, журналы сборки) и собирается большое количество информации из разные мест (30 последних записей журнала изменений, статус в тестируемом выпуске и т. д.). Это очень полезный инструмент, если вы хотите знать, что происходит с конкретным пакетом с исходным

кодом. Более того, после аутентификации вы сможете за один клик подписаться или отписаться от информации о любом пакете.

Вы можете напрямую перейти на веб-страницу конкретного пакета с исходным кодом при помощи URL вида `https://tracker.debian.org/`*пакет-с-исходным-кодом*.

For more in-depth information, you should have a look at its documentation. Among other things, it explains you how to interact with it by email, how to filter the mails that it forwards, how to configure your VCS commit notifications, how to leverage its features for maintainer teams, etc.

4.11 Обзор пакетов разработчика

Веб-портал QA (гарантия качества) доступен по адресу https://qa.debian.org/developer.php, на нём отображается таблица с пакетами одного разработчика (включая те пакеты, у которых в качестве помощников указана группа). Таблица даёт обзор пакетов конкретного разработчика: число ошибок по их важности, список доступных версий в каждом выпуске, статус в тестируемом выпуске и множество ссылок на другую полезную информацию.

Рекомендуется регулярно просматривать эти данные для своих пакетов, так вы не забудете об открытых сообщениях об ошибках и не забудете то, за какие пакеты вы ответственны.

4.12 Установка Debian FusionForge: Alioth

Until Alioth was deprecated and eventually turned off in June 2018, it was a Debian service based on a slightly modified version of the FusionForge software (which evolved from SourceForge and GForge). This software offered developers access to easy-to-use tools such as bug trackers, patch managers, project/task managers, file hosting services, mailing lists, VCS repositories, etc.

For many previously offered services replacements exist. This is important to know, as there are still many references to alioth which still need fixing. If you encounter such references please take the time to try fixing them, for example by filing bugs or when possible fixing the reference.

4.13 Goodies for Debian Members

Benefits available to Debian Members are documented on https://wiki.debian.org/MemberBenefits.

Глава 5

Управление пакетами

Данная глава содержит информацию, связанную с созданием, загрузкой, сопровождением и переносом пакетов.

5.1 Новые пакеты

Если вы хотите создать новый пакет для дистрибутива Debian, вам следует вначале проверить список требующих доработки и будущих пакетов (WNPP). Проверка списка WNPP гарантирует, что в настоящий момент никто не работает над созданием пакетов для данного ПО, и что не будет проделана повторная работа. Прочтите страницу WNPP для получения дополнительной информации.

Допустим, что более никто не работает над вашим будущим пакетом. Далее, вам следует отправить сообщение об ошибке (*Отправка отчётов об ошибках*) в псевдопакете `wnpp` с объяснением вашего плана по созданию нового пакета, ваше сообщение должно в себя включать описание пакета, лицензию будущего пакета и текущий адрес, по которому этот пакет может быть загружен.

You should set the subject of the bug to `ITP:` *foo* `--` *short description*, substituting the *name of the new package* for *foo*. The severity of the bug report must be set to `wishlist`. Please send a copy to `debian-devel@lists.debian.org` by using the X-Debbugs-CC header (don't use CC:, because that way the message's subject won't indicate the bug number). If you are packaging so many new packages (>10) that notifying the mailing list in separate messages is too disruptive, send a summary after filing the bugs to the debian-devel list instead. This will inform the other developers about upcoming packages and will allow a review of your description and package name.

Добавьте запись вида `Closes: #`*nnnnn* в журнал изменений нового пакета для того, чтобы ваше сообщение об ошибке было автоматически закрыто сразу же как только пакет будет установлен в архиве (см. *Когда ошибки исправляются путём новых загрузок*).

Если вы считаете, что по поводу вашего пакета необходимо дать дополнительные объяснения администраторам очереди новых (NEW) пакетов, добавьте их в журнал изменений, отправьте по адресу `ftpmaster@debian.org` ваш ответ на сообщение электронной почты, которое вы получите как сопровождающий после вашей загрузки пакета, либо ответьте на сообщение об отказе, если вы осуществляете загрузку повторно.

Закрывая сообщения об ошибках безопасности, добавляйте номера CVE, а также `Closes: #`*nnnnn*. Это помогает команде безопасности отслеживать уязвимости. Если загрузка осуществляется для того, чтобы исправить ошибку, и если идентификационный номер рекомендации по безопасности ещё не известен, советуем вам во время следующей загрузки изменить запись в журнале изменений

и добавить номер рекомендации. Даже в этом случае добавляйте все доступные указатели на общую информацию из записи оригинального журнала изменений.

Имеется ряд причин, почему мы просим сопровождающих анонсировать их намерения, они приведены ниже:

- Это помогает (потенциальному) сопровождающему получить советы опытных людей, участвующих в списке рассылки, и позволяет этим людям узнать, работает ли уже кто-нибудь над созданием пакета для данного ПО.
- Это позволяет другим людям, которые думают о работе над созданием пакета, знать, что кто-то уже начал этим заниматься, и поэтому можно объединить усилия.
- Это позволяет остальным сопровождающим знать о данном пакете больше, чем то, что содержится в одной строке описания и обычной записи из журнала изменений вида „Initial release“, которые публикуются в `debian-devel-changes@lists.debian.org`.
- Это помогает людям, которые постоянно пользуются `нестабильным` выпуском (и являются теми, кто первый широко тестирует пакеты). Нам следует поощрять этих людей.
- Анонсы дают сопровождающим и другим заинтересованным сторонам лучшее понимание того, что происходит и что является новым в Проекте.

5.2 Запись изменений в пакете

Changes that you make to the package need to be recorded in the `debian/changelog` file, for human users to read and comprehend. These changes should provide a concise description of what was changed, why (if it's in doubt), and note if any bugs were closed. They also record when the packaging was completed. This file will be installed in `/usr/share/doc/`*package*`/changelog.Debian.gz`, or `/usr/share/doc/`*package*`/changelog.gz` for native packages.

Файл `debian/changelog` соответствует определённой структуре, имеющей ряд различных полей. Одно из этих полей, `выпуск`, описывается в *Выбор выпуска*. Дополнительная информация о структуре этого файла может быть найдена в Политике Debian, в разделе с названием `debian/changelog`.

Записи журнала изменений могут использоваться для автоматического закрытия сообщений об ошибках Debian, когда пакет устанавливается в архив. См. *Когда ошибки исправляются путём новых загрузок*.

Обычная запись в журнале изменений какого-либо пакета, содержащего новую версию из основной ветки разработки ПО, выглядит следующим образом:

```
* 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*.

Также см. *Лучшие практики для debian/changelog*.

5.3 Тестирование пакета

До того как вы загрузите ваш пакет, вам следует провести его простое тестирование. Как минимум вам следует попытаться выполнить следующие действия (вам будет нужна более старая версия того же пакета Debian):

- Run `lintian` over the package. You can run `lintian` as follows: `lintian -v` *package-version*`.changes`. This will check the source package as well as the binary package. If you don't understand the output that `lintian` generates, try adding the `-i` switch, which will cause `lintian` to output a very verbose description of the problem.

  Обычно пакет *не* следует загружать в случае, если `lintian` сообщает об ошибках (они начинаются с `E`).

  Для получения дополнительной информации о команде `lintian`, см. *lintian*.

- Также вы можете выполнить `debdiff` (см. *debdiff*), чтобы проанализировать изменения по сравнению с более старой версией (если таковая существует).
- Install the package and make sure the software works in an up-to-date `unstable` system.
- Upgrade the package from an older version to your new version.
- Удалите пакет, затем заново установите его.
- Installing, upgrading and removal of packages can either be tested manually or by using the `piuparts` tool.
- With the package installed, run `adequate <package name>` to test its installed state for policy violations. Ideally, you should automate this step using `autopkgtest` (see *adequate*).
- Скопируйте пакет с исходным кодом в другой каталог и попытайтесь распаковать и собрать его заново. Это позволит проверить то, зависит ли пакет от существующих файлов, которые не были добавлены в сам пакет, а также проверить то, зависит ли он от того, сохраняются ли права доступа в файлах из файла `.diff.gz`.

5.4 Схема пакета с исходным кодом

Имеется два типа пакетов Debian с исходным кодом:

- так называемые `родные` пакеты, для которых нет различия между оригинальным исходным кодом и заплатами Debian
- (более частные) пакеты, для которых имеется оригинальный исходный файл с tarball-архивом и другой файл, содержащий изменения, внесённые Проектом Debian

У родных пакетов пакеты с исходным кодом включают в себя исходный файл контроля (`.dsc`) и исходный архив в виде tarball (`.tar.{gz,bz2,xz}`). Пакет с исходным кодом неродного пакета включает в себя исходный файл контроля, оригинальный исходный файл в виде tarball (`.orig.tar.{gz,bz2,xz}`) и специфичные изменения Debian (`.diff.gz` для формата пакета с исходным кодом "1.0" или `.debian.tar.{gz,bz2,xz}` для формата пакета с исходным кодом "3.0 (quilt)").

Для пакетов в исходном формате "1.0" то, является пакет родным или нет, определялось командой `dpkg-source` во время сборки. Сегодня же рекомендуется явным образом указывать желаемый исходный формат, помещая строку "3.0 (quilt)", либо "3.0 (native)" в файл `debian/source/format`. Остаток настоящего раздела посвящён исключительно неродным пакетам.

The first time a version is uploaded that corresponds to a particular upstream version, the original source tar file must be uploaded and included in the `.changes` file. Subsequently, this very same tar file should be used to build the new diffs and `.dsc` files, and will not need to be re-uploaded.

По умолчанию команды `dpkg-genchanges` и `dpkg-buildpackage` включат оригинальный исходный tar-файл тогда и только тогда, когда текущая запись в журнале изменений содержит версию из основной ветки разработки, которая отличается от предыдущей записи. Это поведение может быть изменено при помощи использования `-sa`, что позволяет всегда включать оригинальный исходный tar-файл, либо `-sd`, что позволяет всегда игнорировать его.

Если оригинальный исходный код не включён в загрузку, оригинальный tar-файлс исходным кодом, используемый `dpkg-source` при создании файла `.dsc` и diff для загрузки *должен* побайтно совпадать с тем файлом, который уже добавлен в архив.

Please notice that, in non-native packages, permissions on files that are not present in the `*.orig.tar.{gz,bz2,xz}` will not be preserved, as diff does not store file permissions in the patch. However, when using source format "3.0 (quilt)", permissions of files inside the `debian` directory are preserved since they are stored in a tar archive.

5.5 Выбор выпуска

Каждая загрузка должна содержать явное указание того, для какого выпуска предназначается данный пакет. Процесс сборки пакета извлекает эту информацию из первой строки файла `debian/changelog` и помещает её в поле `Distribution` файла `.changes`.

Обычно пакеты загружаются в `нестабильный` выпуск. Загрузки в `нестабильный` (`unstable`) или `экспериментальный` (`experimental`) выпуски должны использовать эти названия выпуска в записях журнала изменений; uploads for other supported suites should use the suite codenames, as they avoid any ambiguity.

Фактически, существуют и другие выпуски: *кодовое-имя*`-security`, прочтите *Работа с ошибками, связанными с безопасностью* для получения дополнительной информации.

Нельзя загрузить пакет в несколько выпусков одновременно.

5.5.1 Специальный случай: загрузка в `стабильный` и `предыдущий стабильный` выпуски

Загрузка в `stable` означает, что пакет будет передан в очередь `proposed-updates-new` для проверки управляющими стабильного выпуска, и если пакеты будут одобрены, то они будут установлены в каталог `stable-proposed-updates` архива Debian. Отсюда, в свою очередь, пакеты будут перенесены в `stable` во время формирования следующей редакции выпуска.

Uploads to a supported `stable` release should target their suite name in the changelog, i.e. `trixie` or `bookworm`. You should normally use `reportbug` and the `release.debian.org` pseudo-package to send a *source* `debdiff`, rationale and associated bug numbers to the stable release managers, and await a request to upload or further information.

If you are confident that the upload will be accepted without changes, please feel free to upload at the same time as filing the `release.debian.org` bug. However if you are new to the process, we would recommend getting approval before uploading so you get a chance to see if your expectations align with ours.

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.

- The bug you want to fix in `stable` must be fixed in `unstable` already (and not waiting in NEW or the delayed queue).
- The bug should be of severity "important" or higher.
- Bug meta-data - particularly affected versions - must be up to date.
- Fixes must be minimal and relevant and include a sufficiently detailed changelog entry.
- A source debdiff of the proposed change must be included in your request (not just the raw patches or "a debdiff can be found at $URL").
- The proposed package must have a correct version number (e.g. `...+deb13u1`/`...~deb13u1` for `trixie` or `+deb12u1`/`~deb12u1` for `bookworm`) and you should be able to explain what testing it has had. See the Debian Policy for the version number: https://www.debian.org/doc/debian-policy/ch-controlfields.html#special-version-conventions
- The update must be built in an `stable` environment or chroot (or `oldstable` if you target that).
- Fixes for security issues should be co-ordinated with the security team, unless they have explicitly stated that they will not issue an `DSA` for the bug (e.g. via a "no-dsa" marker in the *Debian Security Tracker*).
- Do not close `release.debian.org` bugs in debian/changelog. They will be closed by the release team once the package has reached the respective point release.

It is recommended to use `reportbug` as it eases the creation of bugs with correct meta-data. The release team makes extensive use of usertags to sort and manage requests and incorrectly tagged reports may take longer to be noticed and processed.

Загрузки в `предыдущие стабильные` выпуски возможны до тех пор, пока эти выпуски не будут добавлены в архив. Те же правила применяются и к `стабильным` выпускам.

In the past, uploads to `stable` were used to address security problems as well. However, this practice is deprecated, as uploads used for Debian security advisories (`DSA`) are automatically copied to the appropriate `proposed-updates` archive when the advisory is released. See *Работа с ошибками, связанными с безопасностью* for detailed information on handling security problems. If the security team deems the problem to be too benign to be fixed through a `DSA`, the stable release managers are usually willing to include your fix nonetheless in a regular upload to `stable`.

5.5.2 Special case: the `stable-updates` suite

Sometimes the stable release managers will decide that an update to stable should be made available to users sooner than the next scheduled point release. In such cases, they can copy the update to the `stable-updates` suite, use of which is enabled by the installer by default.

Initially, the process described in *Специальный случай: загрузка в стабильный и предыдущий стабильный выпуски*. should be followed as usual. If you think that the upload should be released via `stable-updates`, mention this in your request. Examples of circumstances in which the upload may qualify for such treatment are:

- The update is urgent and not of a security nature. Security updates will continue to be pushed through the security archive. Examples include packages broken by the flow of time (c.f. `spamassassin` and the year 2010 problem) and fixes for bugs introduced by point releases.
- The package in question is a data package and the data must be updated in a timely manner (e.g. `tzdata`).
- Fixes to leaf packages that were broken by external changes (e.g. video downloading tools and `tor`).
- Packages that need to be current to be useful (e.g. `clamav`).
- Uploads to `stable-updates` should target their suite name in the changelog as usual, e.g. `trixie`.

Once the upload has been accepted to `proposed-updates` and is ready for release, the stable release managers will then copy it to the `stable-updates` suite and issue a Stable Update Announcement (`SUA`) via the `debian-stable-announce` mailing list.

Any updates released via `stable-updates` will be included in `stable` with the next point release as usual.

5.5.3 Специальный случай: загрузка в `testing/testing-proposed-updates`

Подробности см. в разделе *Прямые обновления тестируемого выпуска* выпуске.

5.6 Загрузка пакета

5.6.1 Source and binary uploads

Each upload to Debian consists of a signed `.changes` file describing the requested change to the archive, plus the source and binary package files that are referenced by the `.changes` file.

If possible, the version of a package that is uploaded should be a source-only changes file. These are typically named `*_source.changes`, and reference the source package, but no binary `.deb` or `.udeb` packages. All of the corresponding architecture-dependent and architecture-independent binary packages, for all architectures, will be built automatically by the build daemons in a controlled and predictable environment (see *wanna-build* for more details).

For many source-only uploads you can use tag2upload instead, which means that you only need to push a signed Git tag to Salsa, instead of generating and signing `.dsc` and `.changes` files yourself. See https://wiki.debian.org/tag2upload.

There are several situations where a source-only upload is not possible.

The first upload of a new source package (see *Новые пакеты*) must include binary packages, so that they can be reviewed by the archive administrators before they are added to Debian.

If new binary packages are added to an existing source package, then the first upload that lists the new binary packages in `debian/control` must include binary packages, again so that they can be reviewed by the archive administrators before they are added to Debian. It is preferred for these uploads to be done via the `experimental` suite.

Uploads that will be held for review in other queues, such as packages being added to the `*-backports` suites, might also require inclusion of binary packages.

The build daemons will automatically attempt to build any `main` or `contrib` package for which the build-dependencies are available. Packages in `non-free` and `non-free-firmware` will not be built by the build daemons unless the package has been marked as suitable for auto-building (see *Отмечаем несвободные пакеты как собираемые автоматически (auto-buildable)*).

The build daemons only install build-dependencies from the `main` archive area. This means that if a source package has build-dependencies that are in the `contrib`, `non-free` or `non-free-firmware` archive areas, then uploads of that package need to include prebuilt binary packages for every architecture that will be supported. By definition this can only be the case for source packages that are themselves in the `contrib`, `non-free` or `non-free-firmware` archive areas.

Bootstrapping a new architecture, or a new version of a package with circular dependencies (such as a self-hosting compiler), will sometimes also require an upload that includes binary packages.

5.6.2 Загрузка на `ftp-master`

To upload a package, you should upload the files (including the signed changes and dsc file) with anonymous ftp to `ftp.upload.debian.org` in the directory /pub/UploadQueue/. To get the files processed there, they need to be signed with a key in the Debian Developers keyring or the Debian Maintainers keyring (see https://wiki.debian.org/DebianMaintainer).

Заметьте, что вам следует переслать файл changes самым последним. В противном случае ваша загрузка может быть отклонены потому, что ПО для сопровождения архива, осуществляющее грамматический разбор файла changes, посчитает, что не все файлы были загружены.

Для загрузки пакетов Вам могут пригодиться пакеты *dupload* или *dput*.Эти удобные программы помогают автоматизировать процесс загрузки пакетов в Debian.

For removing packages or cancelling an upload, please see ftp://ftp.upload.debian.org/pub/UploadQueue/README and the Debian package *dcut*.

Finally, you should think about the status of your package with relation to `testing` before uploading to `unstable`. If you have a version in `unstable` waiting to migrate then it is generally a good idea to let it migrate before uploading another new version. You should also check the *Система отслеживания пакетов Debian* for transition warnings to avoid making uploads that disrupt ongoing transitions.

5.6.3 Задержанные загрузки

It is sometimes useful to upload a package immediately, but to want this package to arrive in the archive only a few days later. For example, when preparing a *Загрузки не-сопровождающим (NMU)*, you might want to give the maintainer a few days to react, while at the same time having the certainty that your upload will eventually take effect if the maintainer does not. A one-shot approach of (i) file bug with a patch (or Merge Request) and (ii) at once upload to DELAYED, is much more efficient than having to wait for days between the two steps.

Загрузка в каталог задержанных пакетов приводит к сохранению пакета в очереди отложенной загрузки. Когда указанное время ожидания истечёт, пакет будет перемещён в обычный каталог

входящих пакетов для его обработки. Это производится путём автоматической загрузки на `ftp.upload.debian.org` в каталог загрузки `DELAYED/`*X*`-day` (*X* может быть в интервале от 0 до 15). 0-дней загружается несколько раз в день на `ftp.upload.debian.org`.

With dput, you can use the `--delayed` *DELAY* parameter to put the package into one of the queues.

Sometimes you'll need to remove a previously uploaded package from the DELAYED queue (e.g. because the package maintainer has asked you so). dcut(1) has examples of how to do so.

5.6.4 Загрузки безопасности

Do **NOT** upload a package to the security upload queue (on `*.security.upload.debian.org`) without prior authorization from the security team. If the package does not exactly meet the team's requirements, it will cause many problems and delays in dealing with the unwanted upload. For details, please see *Работа с ошибками, связанными с безопасностью*.

5.6.5 Другие очереди загрузки

В Европе имеется альтернативная очередь загрузки по адресу ftp://ftp.eu.upload.debian.org/pub/UploadQueue/. Она работает точно так же как и `ftp.upload.debian.org`, но для разработчиков из Европы она может быть более удобной из-за более быстрого доступа.

Кроме того, пакеты можно загружать через ssh на `ssh.upload.debian.org`; файлы должны быть помещены в `/srv/upload.debian.org/UploadQueue`. Эта очередь не поддерживает *Задержанные загрузки*.

5.6.6 Notifications

Сопровождающие архива Debian ответственны за обработку загрузок пакетов. По большей части загрузки обрабатываются автоматически и ежедневно при помощи специальных инструментов для сопровождения архива, `dak process-upload`. Говоря более конкретно, обновление существующих пакетов, предназначенные для `нестабильного` выпуска, обрабатываются автоматически. В других случаях (особенно это касается новых пакетов) помещение загруженного пакета в выпуск осуществляется вручную. Если загрузки обрабатываются вручную, изменение архива может потребовать некоторого времени. Пожалуйста, будьте терпеливы.

В любом случае вы получите уведомление по электронной почте о том, что пакет был добавлен в архив, в этом уведомлении также указаны сообщения об ошибках, которые будут закрыты благодаря данной загрузке. Внимательно изучите это уведомление, проверьте, все ли сообщения об ошибках, которые должны быть закрыты, в нём указаны.

Уведомление об установке также включает в себя информацию о том, в какой раздел был добавлен ваш пакет. Если имеет место несоответствие, то вы получите об этом отдельное сообщение. Читайте ниже.

Заметьте, что если вы загрузили пакет через очереди, служба очередей также отправит вам уведомление по электронной почте.

Also note that new uploads are announced on the *Каналы IRC* channel `#debian-devel-changes`. If your upload fails silently, it could be that your package is improperly signed, in which case you can find more explanations on `ssh.upload.debian.org:/srv/upload.debian.org/queued/run/log`.

5.7 Определение раздела для пакета, подраздела и приоритета

Поля `Section` и `Priority` файла `debian/control` фактически не определяют то, куда в архиве будет помещён ваш пакет, также они не определяют приоритет. Для того, чтобы сохранить общую целостность архива, сопровождающие архива осуществляют контроль над данными полями. Значения в файле `debian/control` в действительности являются лишь подсказками.

Сопровождающие архива отслеживаю канонические разделы и приоритеты пакетов с помощью специального `файла замещения`. Если между `файлом замещения` и полями пакета, определёнными в файле `debian/control`, имеет место несоответствие, то вы получите сообщение о расхождении в момент, когда пакет будет установлен в архив. Вы можете либо исправить ваш файл `debian/control` во время следующей загрузки, либо вы можете захотеть изменить `файл замещения`.

Чтобы изменить раздел, в который был помещён пакет, вам для начала нужно убедиться, что файл `debian/control` в вашем пакете правилен. Далее, отправьте сообщение об ошибке в `ftp.debian.org` с запросом изменения раздела или приоритета для вашего пакета. Используйте тему сообщения на подобие `override: ПАКЕТ1:раздел/приоритет, [...], ПАКЕТX:раздел/приоритет` и добавьте обоснование данного изменения в теле сообщения об ошибке.

Дополнительную информацию о `файлах замещения` см. в dpkg-scanpackages 1 и https://www.debian.org/Bugs/Developer#maintincorrect.

Заметьте, что поле `Section` описывает и раздел, и подраздел, которые описываются в *Разделы*. Если раздел имеет значение main (основной), то его указание должно быть опущено. Список разрешённых подразделов может быть найден в https://www.debian.org/doc/debian-policy/ch-archive.html#s-subsections.

5.8 New upstream versions

To update a package for a new upstream release:

1. Read the upstream changelog, NEWS, and whatever other documentation they may have released with the new version.
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. Port the old Debian packaging to the new version. This basically involves incrementing the `debian/changelog` and merging `debian/patches` from the old package to the new one.
4. Check to see if any bugs have been fixed that are currently open in the BTS. If they have been, close them in the `debian/changelog`.
5. If the patch/merge did not apply cleanly, figure out why. A patch may fail to apply if it's already been applied in the new upstream release, or if the upstream files the patch applies to have been substantially modified (or deleted).
6. If any changes were made to the build system (you'd know from steps 1 and 2), update the `debian/rules` and `debian/control` build dependencies if necessary.
7. Build the new package in an isolated environment, e.g. using `sbuild` or `pbuilder`. This ensures that all required build dependencies are listed in `debian/control` and eliminates the possibility of interference of any third party packages in your system.
8. Verify that the new package builds correctly and if so carry out the checks in *Тестирование пакета*.

5.9 Работа с ошибками

Каждый разработчик должен быть способен работать с системой отслеживания ошибок Debian. Это предполагает знание того, как следует правильно отправлять сообщения об ошибках (см. *Отправка отчётов об ошибках*), как обновлять и упорядочивать их, а также то, как с ними работать и как их закрывать.

Возможности системы отслеживания ошибок описаны в документации по системе отслеживания ошибок для разработчиков. Она включает в себя описание того, как закрывать сообщения об ошиб-

ках, отправлять ответные сообщения, назначать важность и теги, отмечать ошибки как перенаправленные, а также многих других вопросов.

Такие операции как переназначение сообщений об ошибках другим пакетам, объединение отдельных сообщений об ошибках в одну проблему, а также повторное открытие ошибок, если они были закрыты прежде времени, осуществляются при помощи так называемого сервера управляющей почты. Все доступные на этом сервере команды описываются в документации по управляющему серверу системы отслеживания ошибок.

5.9.1 Мониторинг ошибок

Если вы хотите быть хорошим сопровождающим, вам следует периодически проверять в системе отслеживания ошибок Debian (BTS) состояние ваших пакетов. Система отслеживания ошибок содержит все открытые сообщения об ошибках в ваших пакетах. Вы можете проверить их путём просмотра следующей страницы: `https://bugs.debian.org/`*`ваша-учётная-запись`*`@debian.org`.

Сопровождающие взаимодействуют с системой отслеживания ошибок через адреса электронной почты `bugs.debian.org`. Документация по доступным командам может быть найдена в https://www.debian.org/Bugs/, либо, если вы установили пакет `doc-debian`, вы можете посмотреть локальные файлы `/usr/share/doc/debian/bug-*`.

Некоторые разработчики находят полезным получение периодических отчётов об открытых ошибках. Вы можете добавить работу cron подобно следующей, если вы хотите получить еженедельные сообщения с обзором всех открытых сообщений об ошибках в ваших пакетах:

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

Замените *адрес* вашим официальным адресом сопровождающего Debian.

5.9.2 Ответ на ошибки

When responding to bugs, make sure that any discussion you have about bugs is sent to the original submitter of the bug, the bug itself and (if you are not the maintainer of the package) the maintainer. Sending an email to *123*`@bugs.debian.org` will send the mail to the maintainer of the package and record your email with the bug log. If you don't remember the submitter email address, you can use *123*`-submitter@bugs.debian.org` to also contact the submitter of the bug. The latter address also records the email with the bug log, so if you are the maintainer of the package in question, it is enough to send the reply to *123*`-submitter@bugs.debian.org`. Otherwise you should include *123*`@bugs.debian.org` so that you also reach the package maintainer.

Если вы получили сообщение об ошибке, в котором упоминается FTBFS, то это означает Fails to build from source (не удалось собрать из исходного кода). Те, кто занимаются переносами, часто используют данный акроним.

Как только вы закончили работать с сообщением об ошибке (напр., исправили ошибку) отметьте его как `завершённое` (закройте его), отправив сообщение с объяснением по адресу *123*`-done@bugs.debian.org`. Если вы исправили ошибку путём изменения и загрузки пакета, вы можете автоматизировать закрытие сообщения об ошибке как это описано в *Когда ошибки исправляются путём новых загрузок*.

Вам *никогда* не следует закрывать ошибки через отправку команды `close` для сервера ошибок по адресу `control@bugs.debian.org`. Если вы так сделаете, то изначальный автор сообщения об ошибке не получит какой-либо информации о том, почему сообщение об ошибке было закрыта.

5.9.3 Bug housekeeping

Как сопровождающий пакетов вы часто будете находить ошибки в других пакетах, иногда вы будете получать сообщения об ошибках в ваших пакетах, которые в действительности касаются ошибок в других пакетах. Возможности системы отслеживания ошибок описываются в документации по системе отслеживания ошибок для разработчиков Debian. Такие операции как переназначение,

объединение и отметка тегами сообщений об ошибках описываются в документации по управляющему серверу системы отслеживания ошибок. Данный раздел содержит некоторые рекомендации по управлению сообщениями об ошибках в ваших собственных пакетах, эти рекомендации основываются на коллективном опыте разработчиков Debian.

Отправка сообщений об ошибках по поводу проблем, которые вы найдёте в других пакетах, является одной из гражданских обязанностей сопровождающего, см. *Отправка отчётов об ошибках* для получения дополнительной информации. Тем не менее, работа с ошибками в ваших собственных пакетах ещё более важна.

Ниже приведена последовательность шагов, которой можно следовать при работе с сообщением об ошибке:

1. Решите, соответствует присланный отчёт реальной ошибке или нет. Иногда пользователи всего лишь неправильно запускают программу, поскольку они не прочли документацию. Если вы это обнаружите, то просто закройте сообщение об ошибке, предоставив достаточное количество информации, которая позволит пользователю исправить свою проблему (вышлите ссылки на хорошую документацию и так далее). Если вы снова получите то же сообщение об ошибке, задайте себе вопрос, достаточно ли хороша документация, или может быть программа не может обнаружить случаи неправильного использования и выдать информативное сообщение об ошибке. Об этой проблеме следует сообщить автору основной ветки разработки.

   If the bug submitter disagrees with your decision to close the bug, they may reopen it until you find an agreement on how to handle it. If you don't find any, you may want to tag the bug `wontfix` to let people know that the bug exists but that it won't be corrected. Please make sure that the bug submitter understands the reasons for your decision by adding an explanation to the message that adds the `wontfix` tag.

   If this situation is unacceptable, you (or the submitter) may want to require a decision of the technical committee by filing a new bug against the `tech-ctte` pseudo-package with a summary of the situation. Before doing so, please read the recommended procedure.

2. Если ошибка действительно имеет место, но она вызвана другим пакетом, просто переназначьте сообщение об ошибке тому пакету. Если вы не знаете, какому пакету следует переназначить сообщение об ошибке, вам следует попросить помощи в *Каналы IRC* или в `debian-devel@lists.debian.org`. Пожалуйста, проинформируйте сопровождающего того пакета, которому вы переназначаете сообщение об ошибке, например, отправив копию сообщения для системы отслеживания ошибок, содержащего команды для переназначения, по адресу *имя-пакета*`@packages.debian.org`, объясните ему причины в своём сообщении. Заметьте, что простое переназначение *не* пересылается сопровождающим пакета, которому вы переназначаете сообщение об ошибке, поэтому они не будет знать об этом до тех пор, пока они не посмотрят сводную информацию об ошибках в своих пакетах.

   Если ошибка затрагивает работу вашего пакета, вы можете клонировать сообщение об ошибке и переназначить полученную копию тому пакету, который фактически вызывает нежелательное поведение в вашем пакете. В противном случае, ошибка не будет показываться в списке ошибок вашего пакета, что, вероятно, приведёт к тому, что пользователи будут снова и снова сообщать об одной и той же ошибке. Вам следует заблокировать "вашу" ошибку переназначенной, клонированной ошибкой, чтобы засвидетельствовать отношение между ними.

3. Иногда вам также необходимо изменить важность ошибки так, чтобы она соответствовала вашему определению её важности. Люди обычно склонны преувеличивать важность ошибок для того, чтобы их ошибки были быстрее исправлены. Важность некоторых ошибок может понижена до wishlist, если запрашиваемое изменение является скорее косметическим.

4. Если ошибка действительно имеет место, но об этой же проблеме было сообщено кем-то ещё, то два релевантных сообщения должны быть объединены в одно с помощью команды merge. В этом случае если ошибка будет исправлена, все те, кто сообщил о ней, будут уведомлены об этом. (Тем не менее, заметьте, что сообщения, отправленные одному из тех, кто сообщил об ошибке, не будут автоматически перенаправлены остальным пользователям, которые тоже сообщили об ошибке.) Подробности о технической стороне команды merge и родственной ей команды unmerge, см. в документации по управляющему серверу системы отслеживания ошибок.

5. Пользователь, отправивший сообщение об ошибке, мог забыть предоставить какую-то информацию, в этом случае вам следует попросить их предоставить необходимую информацию. Вы можете использовать тег `moreinfo`, чтобы отметить сообщение об ошибке как требующее предоставления дополнительной информации. Более того, если вы не можете воспроизвести ошибку, вы можете пометить сообщение об ошибке тегом `unreproducible`. Это будет означать, что если кто-то может воспроизвести ошибку, то вы просите его предоставить вам дополнительную информацию о том, как воспроизвести эту ошибку. Через несколько месяцев, если эта дополнительная информация не была никем отправлена, сообщение об ошибке может быть закрыто.

6. If the bug is related to the packaging, you just fix it. If you are not able to fix it yourself, then tag the bug as `help`. You can also ask for help on `debian-devel@lists.debian.org` or `debian-qa@lists.debian.org`. See *Координация с разработчиками основной ветки* if it's an upstream problem. If you have the required skills you can prepare a patch that fixes the bug and send it to the author at the same time. Make sure to send the patch to the BTS and to tag the bug as `patch`.

7. Если вы исправили ошибку в своей локальной копии пакета, либо если исправление было загружено в репозиторий системы управления версиями, вы можете отметить сообщение об ошибке тегом `pending`, что будет означать, что ошибка исправлена, и что сообщение об ошибке будет закрыто при следующей загрузке (добавьте пункт `closes:` в ваш файл `changelog`). Это особенно полезно, когда над одним и тем же пакетом работает несколько разработчиков.

8. Once a corrected package is available in the archive, the bug should be closed indicating the version in which it was fixed. This can be done automatically; read *Когда ошибки исправляются путём новых загрузок*.

5.9.4 Когда ошибки исправляются путём новых загрузок

Когда ошибки и проблемы в ваших пакетах будут исправлены, вам как сопровождающему следует закрыть сообщения об этих ошибках. Тем не менее, вам не следует закрывать сообщение об ошибке до того момента, как пакет, содержащий исправление ошибки, будет принят в архив Debian. Следовательно, как только вы получите уведомление о том, что загруженный вами пакет был установлен в архив, вы можете и должны закрыть сообщение об ошибке в системе отслеживания ошибок. Кроме того, сообщение об ошибке должно быть закрыто с указанием правильно версии, содержащей исправление ошибки.

Тем не менее, можно избежать необходимости ручного закрытия сообщений об ошибках после загрузки — просто перечислите ошибки в вашем файле `debian/changelog`, следуя определённым синтаксическим правилам, и ПО по сопровождению архива закроет соответствующие ошибки. Например:

```
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.
```

Технически говоря, следующее регулярное выражение Perl описывает то, как определяются журналы с закрытиями ошибок:

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

We prefer the `closes: #`*XXX* syntax, as it is the most concise entry and the easiest to integrate with the text of the `changelog`. Unless specified differently by the `-v`-switch to `dpkg-buildpackage`, only the bugs closed in the most recent changelog entry are closed (basically, exactly the bugs mentioned in the changelog-part in the `.changes` file are closed).

Исторически загрузки определяются как *Загрузки не-сопровождающим (NMU)* отмечались тегом `fixed`, но не закрывались, но эта практика была прекращена с приходом отслеживания версий. То же самое справедливо и по отношению к тегу `fixed-in-experimental`.

If you happen to mistype a bug number or forget a bug in the changelog entries, don't hesitate to undo any damage the error caused. To reopen wrongly closed bugs, send a `reopen` *XXX* command to the bug tracking system's control address, `control@bugs.debian.org`. To close any remaining bugs that were fixed by your upload, email the `.changes` file to *XXX*`-done@bugs.debian.org`, where *XXX* is the bug number, and put Version: *YYY* and an empty line as the first two lines of the body of the email, where *YYY* is the first version where the bug has been fixed.

Помните, что закрывать ошибки с помощью журнала изменений, как это описано выше, не является чем-то обязательным. Если вы просто хотите закрыть сообщения об ошибках, которые никак не касаются сделанной вами загрузки, закройте ошибку путём отправки вашего объяснения на адрес *XXX*`-done@bugs.debian.org`. **Не** закрывайте ошибки в записи журнала изменений какой-либо версии пакета, если изменения в этой версии пакета не имеют ничего общего с исправление этой ошибки.

Общую информацию о том, как писать журнал изменений, см. в *Лучшие практики для debian/changelog*.

5.9.5 Работа с ошибками, связанными с безопасностью

Из-за того, что ошибки, связанные с безопасностью, обладают скорее некоторой чувствительной природой, с ними следует работать очень внимательно. Команда безопасности Debian существует для того, чтобы координировать эту работу, следить за серьёзными проблемами безопасности, помогать сопровождающим в их работе с проблемами безопасности или исправлять эти проблемы, отправлять рекомендации по безопасности и сопровождать `security.debian.org`.

Если вам становится известно о какой-либо ошибке в пакете Debian, связанной с безопасностью, вне зависимости от того, являетесь вы сопровождающим этого пакета или нет, соберите информацию о проблеме, и сразу же свяжитесь с командой безопасности по электронной почте: `team@security.debian.org`. Если хотите, то можете зашифровать ваше сообщение ключом Debian Security Contact, см. https://www.debian.org/security/faq#contact. **НЕ ЗАГРУЖАЙТЕ** какие-либо пакеты в `стабильный` выпуск, не связавшись с командой безопасности. В качестве полезной информации понимается следующее:

- Известно ли об этой ошибке широкой публике.
- Какие версии пакета подвержены данной ошибке. Проверьте каждую версию, которая в настоящее время поддерживается выпуском Debian, а также `тестируемый` и `нестабильный` выпуски.
- Суть исправления, если оно доступно (заплаты особенно полезны)
- Любые исправленные пакеты, которые вы подготовили самостоятельно (отправьте только debdiff или только файлы `.diff.gz` и `.dsc` и прочтите *Подготовка пакетов для решения проблем безопасности*)
- Любая помощь, которую вы можете предоставить в плане тестирования (уязвимость, тестирование регрессий и т. д.)
- Любая информация, необходимая для рекомендации по безопасности (см. *Рекомендации по безопасности*)

Как сопровождающий пакетов вы ответственны за сопровождение своих пакетов даже в стабильном выпуске. Вы имеете лучшие возможности для оценки заплат и тестирования обновлённых пакетов, поэтому, пожалуйста, внимательно изучите приводимую ниже информацию о том, как подготовить пакеты для работы команды безопасности.

5.9.5.1 Debian Security Tracker

Команда безопасности сопровождает центральную базу данных, систему отслеживания безопасности Debian. Она содержит всю публично доступную информацию о том, что известно о проблемах безопасности: какие пакеты и версии подвержены проблемами, либо были исправлены, а также какие выпуски, стабильный, тестируемый и/или нестабильный, уязвимы. Конфиденциальная информация не добавляется в эту систему.

Вы можете производить поиск по конкретной проблеме или имени пакета. Посмотрите ваш пакет, чтобы узнать, какие проблемы всё ещё открыты. Если вы можете, предоставьте дополнительную информацию об этих проблемах, либо помогите решить их в вашем пакете. Все инструкции доступны на страницах системы отслеживания проблем.

5.9.5.2 Конфиденциальность

В отличии от большей части другой деятельности, которая происходит в Debian, информация о проблемах безопасности иногда держится в секрете в течении некоторого времени. Это позволяет поставщикам ПО координировать раскрытие этой информации, чтобы минимизировать опасность для своих пользователей. Временное закрытие информации об уязвимости зависит от природы проблемы и соответствующего исправления, а также от того, находится ли уже информация об уязвимости в открытом доступе.

Разработчики могут узнать о проблемах безопасности из следующих источников:

- они могут найти упоминание проблемы на публичном форуме (списке рассылки, веб-сайте и т. д.)
- кто-то отправляет отчёт об ошибке
- кто-то информирует через частную почту

В первых двух случаях информация публична, важно подготовить исправление ошибки как можно раньше. Тем не менее, в последнем случае информация об ошибке может не быть публичной. Тогда при работе с проблемой возможны несколько вариантов действий:

- Если проблема безопасности не значительна, иногда нет необходимости скрывать информацию об этой проблеме, следует подготовить исправление и выпустить его.
- Если проблема относится к серьёзным проблемам, желательно сообщить о ней другим поставщикам и скоординировать выпуск. Команда безопасности постоянно находится на связи с различными организациями и индивидами, и заботится о распространении информации и координации.

Если тот, кто сообщает о проблеме, просит не раскрывать её, то в любом случае такой запрос следует уважать, конечно, это не касается информирования команды безопасности с целью подготовки исправления для стабильного выпуска Debian. При отправка конфиденциальной информации команде безопасности, обязательно упомяните о полученной просьбе.

Заметьте, что если требуется сохранить тайну, то вы не можете загрузить исправление в `нестабильный` выпуск (или куда-либо ещё, например, в публичный репозиторий системы управления версиями). Запутывания информации о подробностях изменения не достаточно, поскольку сам код доступен общественности и может быть (и будет) просмотрен остальными людьми.

Однако имеются две причины выпускать информацию даже в том случае, если было запрошено этого не делать: проблема уже была известна в течении некоторого времени, либо проблема или эксплоит доступны публично.

Команда безопасности для общения по поводу чувствительных вопросов использует закодированную при помощи ключа PGP переписку. Подробности см. в ЧаВО команды безопасности.

5.9.5.3 Рекомендации по безопасности

Рекомендации по безопасности выпускаются только для текущего стабильного выпуска, а *не* для `тестируемого` или `нестабильного` выпусков. Рекомендации при их выпуске отправляются в список рассылки `debian-security-announce@lists.debian.org` и размещаются на веб-странице о безопасности. Рекомендации по безопасности пишутся и публикуются командной безопасности. Тем не менее, они вовсе не против того, чтобы сопровождающий предоставил им какую-либо информацию или написал часть текста. В рекомендации по безопасности должна содержаться следующая информация:

- Описание проблемы и её масштаба, включая следующее:
  - Тип проблемы (повышение привилегий, отказ в обслуживании и т. д.)

  - Какие привилегии могут быть получены и кем (если это имеет место)
  - Как эта уязвимость может использоваться
  - Может ли она использоваться удалённо, либо локально
  - Как проблема была исправлена

  Эта информация позволяет пользователям оценить угрозу их системам

- Номера версий подверженных проблеме пакетов
- Номера версий исправленных пакетов
- Информация о том, где можно получить обновлённые пакеты (обычно из архива безопасности Debian)
- References to upstream advisories, CVE identifiers, and any other information useful in cross-referencing the vulnerability

5.9.5.4 Подготовка пакетов для решения проблем безопасности

Вы можете помочь команде безопасности, если предоставите им исправления пакетов, подходящие для рекомендации по безопасности для стабильного выпуска Debian.

When an update is made to the stable release, care must be taken to avoid changing system behavior or introducing new bugs. In order to do this, make as few changes as possible to fix the bug. Users and administrators rely on the exact behavior of a release once it is made, so any change that is made might break someone's system. This is especially true of libraries: make sure you never change the API (Application Program Interface) or ABI (Application Binary Interface), no matter how small the change.

Это означает, что переход на новую версию из основной ветки разработке не является хорошим решением. Вместо этого вам следует осуществить обратный перенос релевантных изменений в версию ПО, которая входит в стабильный выпуск Debian. Как правило, сопровождающие основной ветки разработки помогают сделать это, если это необходимо. Если же нет, то вам может помочь команда безопасности Debian.

В некоторых случаях обратный перенос исправления безопасности невозможен, например, когда должно быть изменено или переписано большое количество исходного кода. Если это имеет место, может потребоваться переход на новую версию основной ветки разработки. Тем не менее, это осуществляется только в крайних случаях, вам всегда следует заранее координировать такой переход с командой безопасности.

С этим де связан и другой важный совет: всегда тестируйте ваши изменения. Если у вас имеется эксплоит, попробуйте использовать его и выяснить, работает ли он на изначальном пакете и действительно ли он не работает на исправленном пакете. Также протестируйте другие, обычные действия, поскольку исправление безопасности может поломать даже кажущиеся несвязанными возможности.

Никогда **НЕ** добавляйте какие-либо изменения в ваш пакет, которые не связаны напрямую с исправлением уязвимости. Это лишь потребует вернуть изменения, это лишь зря потратит время разработчика. Если в вашем пакете имеются другие ошибки, которые вам хотелось бы исправить, подготовьте загрузку в proposed-updates обычным образом уже после того, как будет выпущена рекомендация по безопасности. Механизм обновления безопасности не служит для внесения в ваши пакеты таких изменений, которые в противном случае были бы отвернуты при их загрузке в стабильный выпуск, поэтому, пожалуйста, не пытайтесь этого делать.

Проверьте и протестируйте ваши изменения столько раз, сколько это возможно. Проверьте отличия от предыдущей версии (для этого очень полезны `interdiff` из пакета `patchutils` и `debdiff` из пакета `devscripts`, см. *debdiff*).

Убедитесь, что вы проверили следующее:

- **Укажите верный выпуск** в вашем файле `debian/changelog`: *кодовое-имя*`-security` (напр., `trixie-security`). Не указывайте *выпуск*`-proposed-updates` или `stable`!

- Ваши записи в журнале изменений должны быть информативны и осмысленны. Другие пользователи будут полагаться не них при определении того, была исправлена определённая ошибка или нет. Добавляйте записи `closes:` о любых **ошибках Debian**. Всегда добавляйте внешнюю ссылку, желательно **идентификатор CVE**. Тем не менее, если идентификатор CVE ещё не был назначен, не ждите его, продолжайте процесс. Указать идентификатор можно позже.
- Убедитесь, что **номер версии** верен. Он должен быть больше, чем номер версии текущего пакета, но меньше, чем версии пакетов в более поздних выпусках. Если вы сомневаетесь относительно версии, проверьте её с помощью `dpkg --compare-versions`. Будьте внимательны, не используйте повторно номер версии, который вы использовали для предыдущей загрузки, либо номер, конфликтующий с binNMU. Принято добавлять к номеру версии `+deb`*X*`u1` (где *X* представляет собой главный номер выпуска), напр., `1:2.4.3-4+deb13u1`, для последующей загрузки версию, конечно же, следует увеличить на 1.
- Если исходный код из основной ветки разработки не был ранее загружен на `security.debian.org` (во время предыдущего обновления безопасности), соберите свою загрузку **с полным исходным кодом из основной ветки разработки** (`dpkg-buildpackage -sa`). Если же ранее была произведена загрузка на `security.debian.org`, содержащая ту же самую версию из основной ветки разработки, вы можете осуществить загрузку без добавления исходного кода из основной ветки (`dpkg-buildpackage -sd`).
- Убедитесь, что используется **в точности тот же файл ``*.orig.tar.{gz,bz2,xz}``**, который использовался в обычном архиве, в противном случае переместить исправление безопасности в основной архив будет нельзя.
- Соберите пакет в **чистой системе**, в которой установлен пакеты только из того выпуска, для которого вы собираете свой пакет. Если у вам нет такой системы, вы можете использовать машину debian.org (см. *Машины Debian*), либо настроить chroot (см. *pbuilder* и *debootstrap*).

5.9.5.5 Загрузка исправленного пакета

Do **NOT** upload a package to the security upload queue (on `*.security.upload.debian.org`) without prior authorization from the security team. If the package does not exactly meet the team's requirements, it will cause many problems and delays in dealing with the unwanted upload.

Никогда **НЕ** загружайте ваше исправление в `proposed-updates`, не связавшись с командой безопасности. Пакеты из `security.debian.org` будут скопированы в каталог `proposed-updates` автоматически. Если какой-то пакет с тем же самым или более высоким номером версии уже установлен в архиве, обновление безопасности будет отклонено системой, управляющей архивом. В этом случае стабильный выпуск останется без обновления безопасности.

Once you have created and tested the new package and it has been approved by the security team, it needs to be uploaded so that it can be installed in the archives. For security uploads, the place to upload to is `ftp://ftp.security.upload.debian.org/pub/SecurityUploadQueue/`.

Как только будет принята ваша загрузка в очередь обновления безопасности, пакет будет автоматически собран для всех архитектур и сохранён для проверки командой безопасности.

Uploads that are waiting for acceptance or verification are only accessible by the security team. This is necessary since there might be fixes for security problems that cannot be disclosed yet.

Если член команды безопасности принимает пакет, этот пакет будет установлен в `security.debian.org`, а также предложен для соответствующего каталога *выпуск*`-proposed-updates` на `ftp-master.debian.org`.

5.10 Subscribing to package updates

As a maintainer of a package, you get email notifications for various kinds of events (uploads, BTS messages etc). You can subscribe to receive such notifications also for packages you are not a maintainer of (or packages you maintain collaboratively), by mailing `control@tracker.debian.org` with `subscribe`

<pkgname> either in the subject or the body of the email (see https://qa.pages.debian.net/distro-tracker/usage/messages.html#email-messages for details).

5.11 Перемещение, удаление, переименование, придание статуса осиротевшего, усыновление и повторное введение пакетов

Некоторые действия по управлению архивом в Debian не автоматизированы во время загрузки. Эти процедуры должны быть вручную выполнены сопровождающими. В данной главе содержатся советы о том, что следует делать в этих случаях.

5.11.1 Перемещение пакетов

Иногда для какого-то пакета раздел может быть изменён. Например, пакет из раздела `non-free` (раздел несвободных пакетов) может быть перелицензирован под GPL, в этом случае этот пакет следует переместить в `main` (основной раздел) или `contrib` (раздел ПО, зависящего от несвободного ПО).[1]

Если вам нужно изменить раздел у одного из ваших пакетов, измените управляющую информацию о пакете, затем заново загрузите ваш пакет (подробности см. в Руководстве по политике Debian). Вам следует чётко убедиться, что вы добавили `.orig.tar.{gz,bz2,xz}` в вашу загрузку (даже если вы не загружаете новую версию из основной ветки разработки), либо что этот файл не будет встречаться в новом разделе с остальной частью пакета. Если ваш новый раздел верен, пакет будет перемещён автоматически. Если же пакет не будет перемещён, тогда свяжитесь с управляющими ftp для того, чтобы понять, что же произошло.

Если, с другой стороны, вам необходимо изменить `подраздел` одного из ваших пакетов (напр., „devel“, „admin“), то это уже немного другая процедура. Исправьте подраздел в управляющем файле вашего пакета, затем заново загрузите пакет. Кроме того, вам следует обновить файл отклонений, это описано в *Определение раздела для пакета, подраздела и приоритета*.

5.11.2 Удаление пакетов

If for some reason you want to completely remove a package (say, if it is an old compatibility library which is no longer required), you need to file a bug against `ftp.debian.org` asking that the package be removed; as with all bugs, this bug should normally have normal severity. The bug title should be in the form `RM:` *package [architecture list]* `--` *reason*, where *package* is the package to be removed and *reason* is a short summary of the reason for the removal request. *[architecture list]* is optional and only needed if the removal request only applies to some architectures, not all. Note that the `reportbug` will create a title conforming to these rules when you use it to report a bug against the `ftp.debian.org` pseudo-package.

If you want to remove a package you maintain, you should note this in the bug title by prepending `ROM` (Request Of Maintainer). There are several other standard acronyms used in the reasoning for a package removal; see https://ftp-master.debian.org/removals.html for a complete list. That page also provides a convenient overview of pending removal requests.

Note that removals can only be done for the `unstable`, `experimental` and `stable` distributions. Packages are not removed from `testing` directly. Rather, they will be removed automatically after the package has been removed from `unstable` and no package in `testing` depends on it. (Removals from `testing` are possible though by filing a removal bug report against the `release.debian.org` pseudo-package. See *Удаление из тестируемого выпуска*.)

Имеется одно исключение, при котором явный запрос об удалении не требуется: если двоичный пакет или пакет с исходным кодом более не может быть собран из исходного кода, в этом случае он будет удалён в полуавтоматическом режиме. Для двоичного пакета это означает, что у него более не имеется пакета с исходным кодом, который создаёт данный двоичный пакет; если двоичный пакет

[1] См. Руководство по политике Debian для получения советов о том, к какому разделу относится тот или иной пакет.

не создаётся на некоторых архитектурах, запрос об удалении этого пакета всё равно требуется. Для пакета с исходным кодом это означает, что все двоичные пакеты, на которые ссылается этот пакет, были приняты другим пакетом с исходным кодом.

В вашем запросе об удалении вам следует подробно описать причины, обосновывающие ваше требование. Это позволит избежать нежелательных удалений и отследить то, почему пакет был удалён. Например, вы можете указать имя пакета, которые заменяет удаляемый пакет.

Обычно просят удалить те пакеты, которые сопровождаются самим запрашивающим об удалении. Если вы хотите удалить другой пакет, вам следует получить на это разрешение от сопровождающего этого пакета. Если пакет должен быть признан осиротевшим, если он не имеет сопровождающего, то для начала вам следует обсудить запрос об удалении в `debian-qa@lists.debian.org`. В случае если было принято решение об удалении пакета, вам следует переназначить сообщение об ошибке в пакете `wnpp` и изменить его заголовок на `O:` вместо того, чтобы отправлять новое сообщение об ошибке с запросом об удалении.

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.

If in doubt concerning whether a package is disposable, email `debian-devel@lists.debian.org` asking for opinions. Also of interest is the `apt-cache` program from the `apt` package. When invoked as `apt-cache showpkg` *package*, the program will show details for *package*, including reverse depends. Other useful programs include `apt-cache rdepends`, `apt-rdepends`, `build-rdeps` (in the `devscripts` package) and `grep-dctrl`. Removal of orphaned packages is discussed on `debian-qa@lists.debian.org`.

После удаления пакета следует разобраться с его сообщениями об ошибках. Они должны быть либо переназначены другому пакету в случае, когда фактический код ПО развился в другой пакет (напр., `libfoo12` был удалён из-за того, что `libfoo13` его вытеснил), либо закрыты если это ПО более не является частью Debian. Закрывая сообщения об ошибках, не отмечайте их как исправленный в версиях пакета из предыдущих выпусков Debian, они должны быть отмечены как исправленные в `<наиболее -свежей-версии-в-Debian>+rm`.

5.11.2.1 Удаление пакетов из каталога `Incoming` (каталога входящих пакетов)

In the past, it was possible to remove packages from `incoming`. However, with the introduction of the new incoming system, this is no longer possible.[4] Instead, you have to upload a new revision of your package with a higher version than the package you want to replace. Both versions will be installed in the archive but only the higher version will actually be available in `unstable` since the previous version will immediately be replaced by the higher. However, if you do proper testing of your packages, the need to replace a package should not occur too often anyway.

5.11.3 Замена или переименование пакетов

Если сопровождающие основной ветки разработки решают изменить название своего ПО (либо если вы дали вашему пакету неверное имя), вам необходимо следовать процессу по переименованию пакета, который включает в себя два шага. На первом шаге измените файл `debian/control` так, чтобы в нём было отражено новое имя пакета, а также для того, чтобы устаревшее имя пакета было указано для замены, предоставления и в списке конфликтующих пакетов (см. Руководство по политике Debian). Заметьте, что вам следует добавлять отношение `Provides` только тогда, когда все пакет, зависящие от устаревшего пакета, продолжают работать после переименования. Когда вы загрузите пакет, и он будет перемещён в архив, отправьте сообщение об ошибке в псевдопакете `ftp.debian.org` с просьбой удалить пакет с устаревшим именем (см. *Удаление пакетов*). Не забудьте правильно переназначить сообщения об ошибке, чтобы они отсылали к новому имени пакета.

Вы можете допустить ошибку при создании вашего пакета, тогда вы захотите заменить свой пакет. Единственным способом сделать это является увеличение номера версии и загрузка новой версии пакет. Старая версия как обычно станет устаревшей. Заметьте, что это касается каждой части вашего пакета, включая и исходный код: если вы хотите заменить архив tarball с исходным кодом

[4] Though, if a package still is in the upload queue and hasn't been moved to Incoming yet, it can be removed. (see *Загрузка на ftp-master*)

из основной ветки, вам придётся загрузить ваш пакет с другим номером версии. Проще всего заменить `foo_1.00.orig.tar.gz` на `foo_1.00+0.orig.tar.gz`, либо `foo_1.00.orig.tar.bz2`. Данное ограничение позволяет каждому файлу на ftp иметь уникальное имя, что помогает гарантировать согласованность в сети зеркал.

5.11.4 Придание статуса осиротевшего пакета

If you can no longer maintain a package, you need to inform others, and see that the package is marked as orphaned. You should set the package maintainer to `Debian QA Group <packages@qa.debian.org>` and submit a bug report against the pseudo package `wnpp`. The bug report should be titled `O:` *package* `--` *short description* indicating that the package is now orphaned. The severity of the bug should be set to `normal`; if the package has a priority of standard or higher, it should be set to important. If you feel it's necessary, send a copy to `debian-devel@lists.debian.org` by putting the address in the X-Debbugs-CC: header of the message (no, don't use CC:, because that way the message's subject won't indicate the bug number).

If you just intend to give the package away, but you can keep maintainership for the moment, then you should instead submit a bug against `wnpp` and title it `RFA:` *package* `--` *short description*. `RFA` stands for `Request For Adoption`.

Дополнительная информация доступна на веб-страницах WNPP.

5.11.5 Усыновление пакета

Список пакетов, которым требуются новые сопровождающие, доступен на странице Требующих доработки и будущих пакетов (WNPP). Если вы хотите заняться сопровождением какого-либо пакета из тех, что приведены в списке WNPP, обратитесь к вышеупомянутой странице для получения информации и сведений о процедуре.

It is not OK to simply take over a package without assent of the current maintainer — that would be package hijacking. You can, of course, contact the current maintainer and ask them for permission to take over the package.

However, when a package has been neglected by the maintainer, you might be able to take over package maintainership by following the package salvaging process as described in *Package Salvaging*. If you have reason to believe a maintainer is no longer active at all, see *Работа с неактивными и/или недоступными сопровождающими*.

Complaints about maintainers should be brought up on the developers' mailing list. If the discussion doesn't end with a positive conclusion, and the issue is of a technical nature, consider bringing it to the attention of the technical committee (see the technical committee web page for more information).

Если вы занялись работой над старым пакетом, вам вероятно захочется, чтобы в системе отслеживания ошибок в качестве официального сопровождающего этого пакета были указаны вы. Это будет сделано автоматически как только вы загрузите новую версию с обновлённым полем `Maintainer`, хотя на это может потребоваться несколько часов после выполнения загрузки. Если вы пока не собираетесь загружать новую версию, вы можете использовать информацию из *Система отслеживания пакетов Debian* для получения сообщений об ошибках. Тем не менее, убедитесь, что старый сопровождающий не против того факта, что он будет получать сообщения об ошибках в течении какого-то времени.

To change the maintainer of a pseudo-package, e-mail `owner@lists.debian.org` and ask them to update their `pseudo-packages.maintainers` file. Include the package name, the name and e-mail address for the old and new maintainer, and a reason for the change. Remember to CC the old maintainer on the e-mail.

5.11.6 Повторное добавление пакетов

Пакеты часто удаляются из-за наличия в них критичных для выпуска ошибок, отсутствия сопровождающих, слишком малого числа пользователей или низкого качества. Хотя процесс повторного добавления пакета схож с первоначальной работой над пакетом, вы можете избежать некоторых ловушек, выполнив для начала небольшое историческое исследование.

Для начала вам следует проверить, почему пакет был удалён. Эта информация может быть найдена в сообщении об удалении в разделе новостей на странице системы отслеживания пакетов данного пакета, либо при просмотре журнала удалений. Сообщение об удалении в системе отслеживания ошибок содержит сведения о том, почему пакет был удалён, а также покажет вам то, над чем вам следует поработать для того, чтобы заново добавить пакет. В сообщении может содержаться сведения о том, что вместо повторного добавления пакета лучше всего переключиться на какое-то другое ПО.

Хорошо было бы связаться с предыдущими сопровождающими и выяснить, работают ли они над повторным добавлением пакета, заинтересованы ли они в совместном сопровождении пакета, заинтересованы ли они быть наставником пакета, если это необходимо.

Вам следует сделать всё, что требуется, до того как добавлять новые пакеты (*Новые пакеты*).

You should base your work on the latest packaging available that is suitable. That might be the latest version from `unstable`, which will still be present in the snapshot archive.

Система контроля версий, которая использовалась предыдущим сопровождающим, может содержать полезные изменения, поэтому проверка репозитория может оказаться хорошей идеей. Проверьте, содержит ли файл `control` в предыдущем пакете какие-либо заголовки со ссылками на систему контроля версий для данного пакета, и если да, то существует ли всё ещё этот репозиторий.

Удаление пакета из `нестабильного` выпуска (не из `тестируемого`, `стабильного` или `предыдущего стабильного` выпусков) приводит к закрытию всех сообщений об ошибках, которые связаны с удаляемым пакетом. Вам следует проверить все закрытые сообщения об ошибках (включая архивированные сообщения об ошибках), разархивировать и заново открыть любые закрытые сообщения об ошибках в версии, заканчивающейся на `+rm`, и которые всё ещё актуальны. Любые неактуальные сообщения об ошибках должны быть помечены как исправленные в той версии, в которой они были исправлены, если это, конечно, известно.

Package removals from unstable also trigger marking the package as removed in the *Debian Security Tracker*. Debian members should mark removed issues as unfixed in the security tracker repository and all others should contact the security team to report reintroduced packages.

5.12 Работа на переносом и перенос пакетов

Debian поддерживает всё увеличивающееся число архитектур. Даже если вы не занимаетесь переносом и используете только одну архитектуру, вы как сопровождающий обязаны знать о проблемах, связанных с переносимостью. Следовательно, даже если вы не занимаетесь переносом, вам следует прочесть большую часть данной главы.

Porting is the act of building Debian packages for architectures that are different from the original architecture of the package maintainer's binary package. It is a unique and essential activity. In fact, porters do most of the actual compiling of Debian packages. For instance, when a maintainer uploads a (portable) source package with binaries for the `i386` architecture, it will be built for each of the other architectures, amounting to 8 more builds.

5.12.1 Будьте добры к тем, кто занимается переносом

У тех, кто занимается переносом, трудная и необычная задача, поскольку им необходимо работать с большим объёмом пакетов. В идеале каждый пакет с исходным кодом должен собираться прямо из коробки. К сожалению, зачастую это не так. Данный раздел содержит список „ошибочек“, которые часть совершают сопровождающие Debian — общих проблем, которые ставят в тупик тех, кто занимается переносом, и делают их работу неоправданно сложной.

The first and most important thing is to respond quickly to bugs or issues raised by porters. Please treat porters with courtesy, as if they were in fact co-maintainers of your package (which, in a way, they are). Please be tolerant of succinct or even unclear bug reports; do your best to hunt down whatever the problem is.

По большей части, почти все проблемы, на которые обращают внимание те, кто занимает переносом, вызваны *ошибками, допущенными при создании пакетов*, в пакетах с исходным кодом. Ниже приведён список того, что вам следует проверить или о чём нужно всегда помнить.

1. Make sure that your `Build-Depends` and `Build-Depends-Indep` settings in `debian/control` are set properly. The best way to validate this is to use the `debootstrap` package to create an `unstable` chroot environment (see *debootstrap*). Within that chrooted environment, install the `build-essential` package and any package dependencies mentioned in `Build-Depends` and/or `Build-Depends-Indep`. Finally, try building your package within that chrooted environment. These steps can be automated by the use of the `pbuilder` program, which is provided by the package of the same name (see *pbuilder*).

   Если вы не можете правильно настроить chroot, вам может помочь `dpkg-depcheck` (см. *dpkg-depcheck*).

   Инструкции по настройке сборочных зависимостей см. в Руководстве по политике Debian.

2. Не указывайте в качестве архитектуры отличное от `all` ли `any` значение, если только вы действительно не имеете это в виду. В большинстве случаев сопровождающие не следуют инструкциям Руководства по политике Debian. Установка архитектуры в значение только какой-то одной архитектуры (такой как `i386` или `amd64`) обычно оказывается неправильным.

3. Make sure your source package is correct. Do `dpkg-source -x` *package*`.dsc` to make sure your source package unpacks properly. Then, in there, try building your package from scratch with `dpkg-buildpackage`.

4. Убедитесь, что в вашем пакете с исходным кодом нет файлов `debian/files` или `debian/substvars`. Они должны быть удалены при помощи цели `clean` из `debian/rules`.

5. Make sure you don't rely on locally installed or hacked configurations or programs. For instance, you should never be calling programs in `/usr/local/bin` or the like. Try not to rely on programs being set up in a special way. Try building your package on another machine, even if it's the same architecture.

6. Не используйте при сборке пакета какие-либо уже установленные пакеты (это более конкретный вариант приведённого выше случая). Конечно, бывают и исключения для этого правила, но помните, что любой подобный случай требует ручного вмешательства и не может быть выполнен автоматическими сборщиками пакетов.

7. Если это возможно, не используйте компилятор какой-то конкретной версии. Если это невозможно, то убедитесь, что ваши сборочные зависимости отражают это ограничение, хотя вы, вероятно, просите о чём-то проблематичном, поскольку разные архитектуры иногда основываются на разных компиляторах.

8. Убедитесь, что ваш файл `debian/rules` содержит отдельные цели `binary-arch` и `binary-indep`, как то требуется в Руководстве по политике Debian. Убедитесь, что обе цели работают независимо друг от друга, то есть, что вы можете вызвать одну цель, не вызывая до этого другой. Для проверки этого, попытайтесь запустить `dpkg-buildpackage -B`.

9. When you can't support your package on a particular architecture, you shouldn't use the Architecture field to reflect that (it's also a pain to maintain correctly). If the package fails to build from source, you can just let it be and interested people can take a look at the build logs. If the package would actually build, the trick is to add a `Build-Depends` on `unsupported-architecture [the-not-supported-arch]`. The buildds will not build the package as the build dependencies are not fulfilled on that arch. To prevent building on 32-bits architectures, the `architecture-is-64-bit` build dependency can be used, as `architecture-is-little-endian` can be used to prevent building on big endian systems.

5.12.2 Руководство для загрузок теми, кто занимается переносом

Если пакет собирается из коробки для той архитектуры, на которую он должен быть перенесён, то вам повезло и ваша работа довольно проста. Данный раздел касается этого случая; в разделе описывается то, как собирать и загружать ваш двоичный пакет так, чтобы он был правильно установлен в архив. Если вам требуется внести изменения в пакет для того, чтобы он мог быть

скомпилирован для других архитектур, вам будет нужно выполнить NMU пакета с исходным кодом, поэтому обратитесь к *Когда и как делать NMU*.

При загрузке тем, кто занимается переносом, изменения исходного кода не вносятся. Вам не нужно трогать какие-либо файлы в пакете с исходным кодом. Это предполагает и файл `debian/changelog`.

The way to invoke `dpkg-buildpackage` is as `dpkg-buildpackage -B -m` *porter-email*. Of course, set *porter-email* to your email address. This will do a binary-only build of only the architecture-dependent portions of the package, using the `binary-arch` target in `debian/rules`.

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 Повторная компиляция или только двоичные NMU

Иногда первоначальная загрузка, связанная с переносом, проблематична из-за того, что окружение, в котором был собран пакет, не было достаточно хорошо (устаревшая или не используемая более библиотека, плохой компилятор и т. д.). Тогда вам может потребоваться заново скомпилировать пакет в обновлённом окружении. Тем не менее, в этом случае вам придётся увеличить номер версии пакета, чтобы старый плохой пакет был заменён на новый в архиве Debian (`dak` отказывается устанавливать новые пакеты, если номер их версии не больше уже доступных пакетов).

Вам следует убедиться, что ваша двоичная NMU не приведёт к тому, что пакет перестанет устанавливаться. Это может произойти в случае, если пакет с исходным кодом порождает зависящие и независящие от архитектуры пакеты, взаимные зависимости которые созданы подстановкой переменной dpkg `$(Source-Version)`.

Несмотря на необходимое изменение журнала изменений, это называется двоичной NMU — в этом случае нет необходимости делать пакеты других архитектур устаревшими и заново их компилировать.

Для такой повторной компиляции требуется специальное „магическое“ назначение версий, так чтобы инструменты для сопровождения архива распознали, что хотя даже и имеется новая Debian-версия, соответствующего обновления исходного кода не было. Если вы сделаете это неправильно, сопровождающие архива отклонят вашу загрузку (из-за отсутствия соответствующего исходного кода).

„Магия“ для NMU повторной компиляции включается путём использования суффикса в номере версии пакет, имеющего следующий вид: `b`*номер*. Например, если последняя версия пакета, которую вы заново компилируете, была версией `2.9-3`, ваша двоичная NMU должна иметь версию `2.9-3+b1`. Если последней версией была `3.4+b1` (т. е., родной пакет, для которого в предыдущий раз уже была выполнена NMU с повторной компиляцией), ваша двоичная NMU должна иметь номер версии `3.4+b2`.[2]

Подобно первоначальной загрузке, связанной с переносом, правильный вызов `dpkg-buildpackage` имеет вид `dpkg-buildpackage -B`, что означает, что будут собраны только зависящие от архитектуры части пакета.

5.12.2.2 Когда следует делать NMU, если вы занимаетесь переносом

Те, кто занимаются переносом и выполняют NMU исходного кода, следуют советам из *Загрузки не-сопровождающим (NMU)*, как и те, кто переносом не занимается. Тем не менее, ожидается, что цикл ожидания для NMU исходного кода, выполненной тем, кто занимается переносом, меньше, чем для того, кто переносом не занимается, поскольку те, кто занимается переносом, должны охватить большое количество пакетов. Опять же, ситуация зависит от того, в какой выпуск они хотят осуществить загрузку. Также важно то, является ли данная архитектура кандидатом на включение в следующий стабильный выпуск; управляющие выпуском решают и сообщают о том, какие архитектуры являются такими кандидатами.

[2] В прошло подобные NMU использовали номер третьего уровня в части Debian редакции для обозначения статуса повторной компиляции; тем не менее, этот синтаксис был двусмыслен в случае родных пакетов и не позволяя правильно определять порядок заново скомпилированных NMU, NMU исходного кода и NMU безопасности одного и того же пакета, потому он был отброшен и заменён новым синтаксисом.

Если вы не занимаетесь переносом и выполняете NMU для `нестабильного` выпуска, вам следует придерживаться приведённых выше советов по переносу, но с двумя изменениями. Во-первых, период ожидания принятия — время между тем, когда было сообщено об ошибке в систему отслеживания ошибок, и тем, когда для NMU наступает подходящее время — равен семи дням для тех, кто занимается переносом, и работает над `нестабильным` выпуском. Этот период может быть уменьшен по усмотрению тех, кто занимается переносом, в случае, если проблема является критической и представляет собой серьёзную трудность для процесса переноса пакета. (Помните, это не является Политикой, это лишь негласно принято в сообществе.) Для выполнения загрузки в `стабильный` или `тестируемый` выпуски свяжитесь с соответствующей выпускающей командой.

Во-вторых, те, кто занимается переносом и выполняют NMU исходного кода, должны убедиться, что сообщение об ошибке, отправленное в систему отслеживания ошибок, имеет важность `serious` или выше. Это гарантирует, что отдельный пакет с исходным кодом может использоваться для компиляции пакета для каждой поддерживаемой Debian на момент выпуска архитектуры. Важно, что у нас имеется одна версия двоичного пакета и пакета с исходным кодом для всех архитектур для того, чтобы соответствовать множеству лицензий.

Те, кто занимаются переносом, должны попытаться избежать заплат, которые просто обходят ошибки в текущей версии окружения компиляции, ядра или libc. Иногда такие клуджи не могут помочь. Если вам нужно обойти ошибки компилятора или чего-то подобного, убедитесь, что вы выполнили корректно `#ifdef` для своей работы; также, документируйте ваши клуджи, чтобы люди знали, как удалить их когда внешние проблемы будут исправлены.

У тех, кто занимается переносом, также имеется неофициальное место, куда они могут поместить результаты своей работы во время периода ожидания. Это помогает остальным, кто занимается переносом, использовать вашу работу даже во время периода ожидания. Конечно, такие места не являются официальными и не имеют какого-либо специального статуса, поэтому будьте внимательны.

5.12.3 Инфраструктура переноса и автоматизация

Имеется специальная инфраструктура и некоторые инструменты, необходимые для автоматизации переноса пакетов. Данный раздел сдержит краткий обзор автоматизации и переноса с помощью данных инструментов; дополнительную информацию см. в документации пакетов или по другим ссылкам.

5.12.3.1 Списки рассылки и веб-страницы

Веб-страницы, содержащие информацию о статусе каждого переноса, можно найти по адресу: https://www.debian.org/ports/.

У каждого переноса Debian имеется свой список рассылки. Список списков рассылки различных проектов по переносу может быть найден по адресу: https://lists.debian.org/ports.html. Эти списки рассылки используются для координации работы и связи пользователей с теми, кто работает над переносом.

5.12.3.2 Инструменты переноса

Описания некоторых инструментов переноса могут быть найдены в *Инструменты для переноса*.

5.12.3.3 `wanna-build`

The `wanna-build` system is used as a distributed, client-server build distribution system. It is usually used in conjunction with build daemons running the `buildd` program. `Build daemons` are `slave` hosts, which contact the central `wanna-build` system to receive a list of packages that need to be built.

`wanna-build` is not yet available as a package; however, all Debian porting efforts are using it for automated package building. The tool used to do the actual package builds, `sbuild`, is available as a package; see its description in *sbuild*. Please note that the packaged version is not the same as the one used on build daemons, but it is close enough to reproduce problems.

Most of the data produced by `wanna-build` that is generally useful to porters is available on the web at https://buildd.debian.org/. This data includes nightly updated statistics, queueing information and logs for build attempts.

Мы очень гордимся этой системой, поскольку у неё так много возможных пользователей. Независимые от разработки группы могут использовать данную систему для подготовки различных вариантов Debian, которые иногда могут быть интересы и более широкому кругу пользователей (например, вариант Debian, собранный с поддержкой проверки связывания средствами `gcc`). Кроме того, она позволяет довольно быстро заново собрать целый выпуск Debian.

Связаться с командой wanna-build, ответственной за службы buildd, можно по адресу: `debian-wb-team@lists.debian.org`. Для того, чтобы определить, с кем (команда wanna-build, команда выпуска) и как (почта, система отслеживания ошибок) следует связаться в конкретном случае, см. https://lists.debian.org/debian-project/2009/03/msg00096.html.

При запросе binNMU или возвратов (повторов после неудачной сборки) используйте описанный в https://release.debian.org/wanna-build.txt формат.

5.12.4 Если ваш пакет *не* может быть перенесён

Некоторые пакеты всё равно имеют проблемы со сборкой и/или с работой на некоторых поддерживаемых Debian архитектурах, они вообще не могут быть перенесены, либо для их переноса требуется слишком много времени. Примером этого является пакет, который конкретно связан с SVGA (доступен только для `i386` и `amd64`), либо использует другие специфические возможности оборудования, которые совсем не поддерживаются на всех архитектурах.

Для того, чтобы сломанные пакеты не были загружены в архив и не потратили зря время работы buildd, вам следует выполнить несколько вещей:

- Во-первых, убедитесь, что ваш пакет *действительно* не может быть собран на архитектурах, которые он не может поддерживать. Для достижения этого имеется несколько способов. Предпочтительный способ состоит в том, чтобы иметь небольшой тестовый набор для использования во время сборки, который будет проверять функциональность пакета и который будет завершаться неудачей в случае, если пакет не работает. Отличной идеей является и то, что это помешает (некоторым) сломанным загрузкам на все архитектуры, а также позволит пакету собираться сразу как только требуемая функциональность будет доступна.

  Кроме того, если вы убеждены, что список поддерживаемых архитектур вполне постоянен, вам следует изменить `any` на список поддерживаемых архитектур в `debian/control`. При этом способе сборка опять же завершиться неудачно, пользователю будет сообщено об этом, даже хотя попытки сборки не было.

- Для того, чтобы ПО для автоматической сборки не пыталось без необходимости на то собрать ваш пакет, в `Packages-arch-specific` должен быть включён список, используемый сценарием `wanna-build`. Текущая версия доступна как https://wiki.debian.org/PackagesArchSpecific; с кем можно связаться по поводу изменений см. в начале файла.

Please note that it is insufficient to only add your package to `Packages-arch-specific` without making it fail to build on unsupported architectures: A porter or any other person trying to build your package might accidentally upload it without noticing it doesn't work. If in the past some binary packages were uploaded on unsupported architectures, request their removal by filing a bug against `ftp.debian.org`.

5.12.5 Отмечаем несвободные пакеты как собираемые автоматически (autobuildable)

By default packages from the `non-free` and `non-free-firmware` sections are not built by the autobuilder network (mostly because the license of the packages could disapprove). To enable a package to be built, you need to perform the following steps:

1. Проверьте, законно ли и возможно ли технически автоматически собирать данный пакет;
2. Добавьте `XS-Autobuild: yes` в заголовок файла `debian/control`;

3. Send an email to `non-free@buildd.debian.org` and explain why the package can legitimately and technically be auto-built.

5.13 Загрузки не-сопровождающим (NMU)

Каждый пакет имеет одного или нескольких сопровождающих. Обычно это люди, которые работают над пакетом и выполняют загрузки новых версий. В некоторых ситуациях бывает полезно, если и другие разработчики смогут загрузить новую версию, например, если они хотят исправить ошибку в пакете, сопровождением которого они не занимаются, и если сопровождающий нуждается в помощи для решения проблем с пакетом. Такие загрузки называются *загрузки не-сопровождающими, Non-Maintainer Uploads (NMU)*.

5.13.1 Когда и как делать NMU

До выполнения NMU, ответьте на следующие вопросы:

- Have you geared the NMU towards helping the maintainer? As there might be disagreement on the notion of whether the maintainer actually needs help or not, the DELAYED queue exists to give time to the maintainer to react and has the beneficial side-effect of allowing for independent reviews of the NMU diff.
- Does your NMU really fix bugs? ("Bugs" means any kind of bugs, e.g. wishlist bugs for packaging a new upstream version, but care should be taken to minimize the impact to the maintainer.) Using NMUs to make changes that are likely to be non-consensual is discouraged.
- As more specific examples, the following changes are generally considered acceptable, unless there are good reasons for not following those practices in a particular package: using the latest released debhelper compatibility level; using dh; using 3.0 (quilt); using lintian-brush.
- Даёте ли вы достаточное количество времени сопровождающему? Когда в системе отслеживания ошибок было прислано сообщение об ошибке? Человек может быть занят одну или две недели, в этом не ничего необычного. Так ли серьёзная ошибка, что её необходимо исправить прямо сейчас, можно ли подождать ещё несколько дней?
- Насколько вы уверены в своих изменениях? Помните клятву Гиппократа: "Не навреди." Лучше оставить пакет даже с самой серьёзной ошибкой, чем применять к нему неработающую заплату, либо заплату, которая скорее скрывает ошибку, а не решает её. Если вы не уверены на 100% в том, что вы делаете, лучше будет попросить совета у других. Помните, что если вы что-то сломаете во время NMU, многие люди будут недовольны.
- Выразили ли вы ясно ваше намерение сделать NMU, по меньшей мере в системе отслеживания ошибок? Если вы не получили какого-либо ответа, то попытайтесь связаться с сопровождающим другими способами (напишите сообщение на адрес электронной почты сопровождающего, на его частный адрес электронной почты, через IRC).
- Если сопровождающий обычно активен и отзывчив, попытались ли вы с ним связаться? Вообще же предпочтительно, чтобы сопровождающие самостоятельно решали проблемы в своих пакетах, нужно дать им возможность проверить вашу заплату и при необходимости исправить её, поскольку они скорее всего лучше осведомлены о потенциальных проблемах, которые могут быть упущены в ходе NMU. Часто время каждого используется значительно лучше, если у сопровождающего будет возможность самому загрузить исправление.

Если вы выполняете NMU, вам для начала следует убедиться, что ваше намерение сделать NMU ясно и понятно. Затем вам необходимо отправить заплату с различиями между текущим пакетом и предполагаемой NMU в системе отслеживания ошибок. Сценарий `nmudiff` из пакета `devscripts` может вам в этом помочь.

While preparing the patch, you had better be aware of any package-specific practices that the maintainer might be using. Taking them into account reduces the burden of integrating your changes into the normal package workflow and thus increases the chances that integration will happen. A good place to look for possible package-specific practices is `debian/README.source`.

Если у вас нет каких-либо веских причин не давать сопровождающему время для самостоятельной работы, вам следует дать последнему это время (например, загрузив пакет в очередь `DELAYED`). Для задержек рекомендуется использовать следующие значения:

- Загрузка исправлений критичных для выпуска ошибок, о которых было сообщено более 7 дней назад, сопровождающий не проявлял активности в течении 7 дней, нет никакого указания на то, что исправление находится в стадии подготовки: 0 дней
- Загрузка исправления только критичных для выпуска ошибок, о которых было сообщено более 7 дней назад: 2 дня
- Исправление только критичных для выпуска ошибок и важных ошибок: 5 дней
- Other NMUs: 15 days

Эти задержки являются лишь примерами. В некоторых случаях, таких как загрузка исправлений безопасности, либо исправлений тривиальных ошибок, блокирующий перемещение пакета, желательно, чтобы исправленный пакет попал в `нестабильный выпуск` как можно скорее.

Sometimes, release managers decide to encourage NMUs with shorter delays for a subset of bugs (e.g release-critical bugs older than 7 days). Also, some maintainers list themselves in the Low Threshold NMU list, and accept that NMUs are uploaded without delay. But even in those cases, it's still a good idea to give the maintainer a few days to react before you upload, especially if the patch wasn't available in the BTS before, or if you know that the maintainer is generally active.

After you upload an NMU, you are responsible for the possible problems that you might have introduced. You must keep an eye on the package (*Subscribing to package updates* is a good way to achieve this).

Не разрешается бездумно выполнять NMU. Если вы выполняете NMU, и ясно, что сопровождающие активны и приняли бы заплату немного позже, либо если вы игнорируете рекомендации, приведённые в данном документе, ваша загрузка может привести к конфликту с сопровождающим. Вам всегда следует быть готовым к отстаиванию своего решения относительно любой загрузки NMU, которую вы выполняете.

5.13.2 NMU и файл `debian/changelog`

Just like any other (source) upload, NMUs must add an entry to `debian/changelog`, telling what has changed with this upload. The first line of this entry must explicitly mention that this upload is an NMU, e.g.:

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

Присваивание версий при выполнении NMU разница для родных и неродных пакетов.

Если пакет является родным пакетом (без номера ревизии Debian в номере версии), версия должны совпадать с версией загрузки её последним сопровождающим, плюс `+nmu`X, где X представляет собой счётчик, начинающийся с `1`. Если последняя загрузка также была NMU, то счётчик следует увеличить. Например, если текущая версия пакета равна `1.5`, то загрузка NMU должна получить версию `1.5+nmu1`.

Если пакет не является родным пакетом, вам следует добавить минорный номер версии к части о ревизии Debian номера версии (та часть, которая идёт после последнего знака тире). Этот дополнительный номер должен начинаться с `1`. Например, если текущей версией является `1.5-2`, то загрузка NMU должна иметь версию `1.5-2.1`. Если в ходе NMU создаётся пакет для новой версии из основной ветки разработки, то номер ревизии Debian устанавливается в `0`, например, `1.6-0.1`.

В обоих случаях если последняя загрузка также была загрузкой NMU, счётчик должен быть увеличен. Например, если текущей версией является `1.5+nmu3` (родной пакет, для которого уже была выполнена загрузка NMU в прошлый раз), загрузка NMU должна получить версию `1.5+nmu4`.

Специальная схема версий требуется для того, чтобы избежать срыва работы сопровождающего, поскольку использование целого числа для ревизии Debian потенциально может привести к конфликту с загрузкой, которая уже находится в стадии подготовки самим сопровождающим в то время, как вы выполняете NMU, либо даже уже включена в очередь NEW на ftp. Кроме того, это

полезно для визуального выделения того, что пакет в архиве был подготовлен не тем, кто является его официальным сопровождающим.

Если вы загружаете пакет в тестируемый или стабильный выпуски, иногда вам необходимо выполнить "разветвление" древа номера версии. Это делается, например, в случае подготовки загрузок с исправлениями безопасности. Для этого следует использовать версию вида `+deb`*X*`u`*Y*, где *X* представляет собой мажорный номер версии, а *Y* является счётчиком, начинающимся с `1`. Например, поскольку `trixie` (Debian 13) является стабильным выпуском, загрузка NMU для исправления проблем безопасности стабильного выпуска для пакета, имеющего версию `1.5-3` будет иметь версию `1.5-3+deb13u1`, а загрузка NMU с исправлением безопасности для `forky` будет иметь версию `1.5-3+deb14u1`.

5.13.3 Использование очереди DELAYED/

Ожидание ответа на ваш запрос разрешения выполнить NMU не эффективно, так как это предполагает, что тот, кто выполняет NMU, должен будет отвлечься от проблемы, а затем снова вернутся к ней. Очередь `DELAYED` (см. *Задержанные загрузки*) позволяет разработчику, занимающемуся NMU, в то же самое время выполнять все необходимые задачи. Например, вместо того, чтобы сообщить сопровождающему о том, что вы собираетесь загрузить обновлённый пакет в течении 7 дней, вам следует загрузить пакет в `DELAYED/7` и сообщить сопровождающему, что у него имеется 7 дней для того, чтобы отреагировать на это. В течении этого времени сопровождающий может попросить вас ещё немного задержать загрузку, либо отменить её.

You can cancel your upload using *dcut*. In case you uploaded `foo_1.2-1.1_all.changes` to a `DELAYED` queue, you can run `dcut cancel foo_1.2-1.1_all.changes` to cancel your upload. The `.changes` file does not need to be present locally as you instruct *dcut* to upload a command file removing a remote filename. The `.changes` file name is the same that you used when uploading.

Очередь `DELAYED` не должна использоваться для дополнительного давления на сопровождающего. В частности, важно, что у вас имеется возможность отметить или задержать загрузку ещё немного до того момента, как срок изначальной задержки закончиться, так как сопровождающий не может сам отметить вашу загрузку.

Если вы выполняете NMU в `DELAYED`, а сопровождающий обновляет пакет до того, как закончится срок задержки, ваша загрузку будет отклонена, поскольку в архиве уже будет доступна более новая версия. В идеале сопровождающий позаботится о включении предлагаемых вами изменений (или по меньшей мере о решении проблем, для решения который предназначаются ваши изменения) в своей загрузке.

5.13.4 NMU с точки зрения сопровождающего

Когда кто-то выполняет NMU для вашего пакета, то это означает, что он хочет помочь вам поддерживать пакет в хорошей форме. Это позволяет пользователям быстрее получать исправленные пакеты. Вы можете попросить того, кто выполнил NMU, стать помощником сопровождающего для данного пакета. Когда кто-то выполнил NMU для вашего пакета, это вовсе не плохо; это лишь означает, что пакет интересен достаточному количеству людей, которые готовы работать над ним.

Для подтверждения NMU, добавьте предлагаемые изменения и запись журнала изменений в вашу собственную загрузку. Если вы не подтвердите NMU, добавив запись из журнала изменений NMU в ваш журнал изменений, сообщения об ошибках в системе отслеживания ошибок останутся закрытыми, но будут указаны как актуальные для версии пакета, которую загрузите вы.

Note that if you ever need to revert a NMU that packages a new upstream version, it is recommended to use a fake upstream version like *CURRENT*`+really`*FORMER* until one can upload the latest version again. More information can be found in 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 NMU исходного кода и двоичные NMU (binNMU)

Полностью NMU называется *NMU исходного кода*. Есть и другой тип NMU, а именно *двоичные NMU* или *binNMU*. BinNMU также представляет собой загрузку пакета тем, что не является его сопровождающим. Тем не менее, это загрузка исключительно двоичного кода.

При обновлении библиотеки (либо другой зависимости) пакеты, использующие её вероятно потребуется собрать заново. Поскольку изменения исходного кода не требуются, используется тот же самый пакет с исходным кодом.

BinNMU обычно выполняются на buildd по инструкции wanna-build. В файл `debian/changelog` добавляется запись, объясняющая то, почему потребовалась загрузка, номер версии увеличивается в соответствии с тем, как это описано в *Повторная компиляция или только двоичные NMU*. Эту запись не следует включать в следующую загрузку.

Buildd загружает пакеты для своей архитектуры в архив как двоичные загрузки. Строго говоря, они являются binNMU. Тем не менее, обычно они не называются NMU, они не добавляют запись в `debian/changelog`.

5.13.6 NMU и загрузки командой контроля качества

NMUs are uploads of packages by somebody other than their assigned maintainer. There is another type of upload where the uploaded package is not yours: QA uploads. QA uploads are uploads of orphaned packages.

Загрузки, выполняемые командой контроля качества, очень похожи на загрузки обычных сопровождающих: они могут исправлять всё, что угодно, даже незначительные проблемы; присвоение номера версии происходит обычным путём, также нет необходимости использовать задержку при загрузке. Отличие состоит в том, что вы не указаны в полях `Maintainer` и `Uploader` данного пакета. Кроме того, запись журнала изменений при загрузке, выполняемой командой контроля качества, содержит специальную первую строку:

```
* QA upload.
```

Если вы хотите выполнить NMU, и у вас складывается впечатление, что сопровождающий не активен, вам следует проверить, является ли данный пакет осиротевшим пакетом (эта информация отображается на странице пакета в системе отслеживания пакетов). При выполнении первой загрузки командой контроля качества в качестве сопровождающего устанавливается `Debian QA Group <packages@qa.debian.org>`. Для осиротевших пакетов, загрузка которых командой контроля качества ещё не выполнялась, в качестве сопровождающего имеют своего старого сопровождающего. Имеется список таких пакетов: https://qa.debian.org/orphaned.html.

Instead of doing a QA upload, you can also consider adopting the package by making yourself the maintainer. You don't need permission from anybody to adopt an orphaned package; you can just set yourself as maintainer and upload the new version (see *Усыновление пакета*).

5.13.7 NMU и командные загрузки

Sometimes you are fixing and/or updating a package because you are member of a packaging team (which uses a mailing list as `Maintainer` or `Uploader`; see *Совместное сопровождение*) but you don't want to add yourself to `Uploaders` because you do not plan to contribute regularly to this specific package. If it conforms with your team's policy, you can perform a normal upload without being listed directly as `Maintainer` or `Uploader`. In that case, you should start your changelog entry with the following line:

```
* Team upload.
```

5.14 Package Salvaging

Package salvaging is the process by which one attempts to save a package that, while not officially orphaned, appears poorly maintained or completely unmaintained. This is a weaker and faster procedure than orphaning a package officially through the powers of the MIA team. Salvaging a package is not

meant to replace MIA handling, and differs in that it does not imply anything about the overall activity of a maintainer. Instead, it handles a package maintainership transition for a single package only, leaving any other package or Debian membership or upload rights (when applicable) untouched.

Note that the process is only intended for actively taking over maintainership. Do not start a package salvaging process when you do not intend to maintain the package for a prolonged time. If you only want to fix certain things, but not take over the package, you must use the NMU process, even if the package would be eligible for salvaging. The NMU process is explained in *Загрузки не-сопровождающим (NMU)*.

Another important thing to remember: It is not acceptable to hijack others' packages. If followed, this salvaging process will help you to ensure that your endeavour is not a hijack but a (legal) salvaging procedure, and you can counter any allegations of hijacking with a reference to this process. Thanks to this process, new contributors should no longer be afraid to take over packages that have been neglected or entirely forgotten.

The process is split into two phases: In the first phase you determine whether the package in question is *eligible* for the salvaging process. Only when the eligibility has been determined you may enter the second phase, the *actual* package salvaging.

For additional information, rationales and FAQs on package salvaging, please visit the Salvaging Packages page on the Debian wiki.

5.14.1 When a package is eligible for package salvaging

A package becomes eligible for salvaging when it has been neglected by the current maintainer. To determine that a package has really been neglected by the maintainer, the following indicators give a rough idea what to look for:

- NMUs, especially if there has been more than one NMU in a row.
- Bugs filed against the package do not have answers from the maintainer.
- Upstream has released several versions, but despite there being a bug entry asking for it, it has not been packaged.
- There are QA issues with the package.

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 How to salvage a package

If and *only* if a package has been determined to be eligible for package salvaging, any prospective maintainer may start the following package salvaging procedure.

1. Open a bug with the severity "important" against the package in question, expressing the intent to take over maintainership of the package. For this, the title of the bug should start with `ITS: package-name`[3]. You may alternatively offer to only take co-maintenance of the package. When you file the bug, you must inform all maintainers, uploaders and if applicable the packaging team explicitly by adding them to `X-Debbugs-CC`. Additionally, if the maintainer(s) seem(s) to be generally inactive, please inform the MIA team by adding `mia@qa.debian.org` to `X-Debbugs-CC` as well. As well as the explicit expression of the intent to salvage, please also take the time to document your assessment of the eligibility in the bug report, for example by listing the criteria you've applied and adding some data to make it easier for others to assess the situation.
2. In this step you need to wait in case any objections to the salvaging are raised; the maintainer, any current uploader or any member of the associated packaging team of the package in question

[3] ITS is shorthand for *"Intend to Salvage"*

may object publicly in response to the bug you've filed within `21 days`, and this terminates the salvaging process.

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. After the 21 days delay, if no answer has been sent to the bug from the maintainer, one of the uploaders or team, you may upload the new release of the package into the `DELAYED` queue with a minimum delay of `seven days`. You should close the salvage bug in the changelog and you must also send an nmudiff to the bug ensuring that copies are sent to the maintainer and any uploaders (including teams) of the package by `CC'ing` them in the mail to the BTS.

   During the waiting time of the `DELAYED` queue, the maintainer can accept the salvaging, do an upload themselves or (ask to) cancel the upload. The latter two of these will also stop the salvaging process, but the maintainer must reply to the salvaging bug with more information about their action.

5.15 Совместное сопровождение

Совместное сопровождение представляет собой термин, описывающий разделение обязанностей по сопровождению пакета Debian между несколькими людьми. Эта совместная работа почти всегда является отличной идеей, поскольку её результатом обычно является более высокое качество пакета и более быстрое исправление ошибок. Настоятельно рекомендуется, чтобы пакет с приоритетом `standard`, а также пакеты, являющиеся частью базового набора пакетов, имели несколько сопровождающих.

Обычно имеется главный сопровождающий и один или больше помощников. Главный сопровождающий является тем, чьё имя указано в поле `Maintainer` файла `debian/control`. Помощники сопровождающего — это все остальные сопровождающие, обычно они указаны в поле `Uploaders` файла `debian/control`.

В наиболее простом виде процесс добавления новых помощников сопровождающий крайне лёгок:

- Set up the co-maintainer with access to the sources you build the package from. Generally this implies you are using a network-capable version control system, such as `Git`. Salsa (see *salsa.debian.org: Git repositories and collaborative development platform*) provides Git repositories, amongst other collaborative tools.
- Добавьте имя и адрес электронной почты вашего помощника в поле `Uploaders` из первого раздела файла `debian/control`.

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

- The co-maintainers should subscribe themselves to the appropriate source package (see *Subscribing to package updates*).

Другой формой совместного сопровождения является командное сопровождение, которое рекомендуется в случае если вы сопровождаете несколько пакетов вместе группой одних и тех же разработчиков. В этом случае на поля `Maintainer` и `Uploaders` каждого пакеты следует обратить особое внимание. Рекомендуется выбрать одну из двух следующих схем:

1. Поместите в поле `Maintainer` члена команды, который будет ответственен за данный пакет. В поле `Uploaders` поместите адрес списка рассылки, а также тех членов команды, которые также будут следить за пакетов.

2. Put the mailing list address in the `Maintainer` field. In the `Uploaders` field, put the team members who care for the package. In this case, you must make sure the mailing list accepts bug reports without any human interaction (like moderation for non-subscribers).

В любом случае не следует помещать всех членов команды в поле `Uploaders`. Это приводит к путанице — в обзорный список пакетов разработчика (см. *Обзор пакетов разработчика*) попадают пакеты, о который данный разработчик фактически не заботится, и создаёт у пользователей ложное чувство того, что пакет хорошо сопровождается. По той же самой причине члены команды не должны добавлять себя в поле `Uploaders` только потому, что они загрузили данный пакет один раз, они могут выполнить "командную загрузку" (см. *NMU и командные загрузки*). И наоборот, не следует оставлять пакет только с одним адресом списка рассылки в поле `Maintainer`, когда поле `Uploaders` пусто.

5.16 Тестируемый выпуск

5.16.1 Основы

Обычно пакеты устанавливаются в `тестируемый` выпуск после того, как они пройдут некоторое тестирование в `нестабильном` выпуске.

На всех архитектурах должна быть доступна одна и та же версия пакета, у пакета не должно быть зависимостей, которые помешают установить этот пакет; также они не должны иметь на момент установки в `тестируемый` выпуск известных критичных для выпуска ошибок. Таким образом, `тестируемый` выпуск всегда должен быть близок кандидату на выпуск. Подробности см. ниже.

5.16.2 Обновления из нестабильного выпуска

Сценарии, обновляющие `тестируемый` выпуск, запускаются дважды каждый день, сразу же после установки обновлённых пакетов; эти сценарии имеют имя `britney`. Они создают файлы `Packages` для `тестируемого` выпуска, но делают они это разумным способом; они пытаются избежать противоречивости и использовать только те пакеты, которые не имеют ошибок.

Включение пакета из `нестабильного` выпуска выполняется в соответствии со следующими условиями:

- The package must have been available in `unstable` for a certain number of days, see *Selecting the upload urgency*. Please note that the urgency is sticky, meaning that the highest urgency uploaded since the previous `testing` transition is taken into account;
- Пакет не должен иметь новых ошибок, критичных для выпуска (то есть, критичных для выпуска ошибок в версии пакета, доступной в `нестабильном` выпуске, но отсутствующих в версии из `тестируемого` выпуска);
- Пакет должен быть доступен на всех архитектурах, на которых он ранее был собран в `нестабильном` выпуске. *Утилита dak ls* может помочь проверить эту информацию;
- Пакет не должен ломать зависимости какого-либо уже доступного в `тестируемом` выпуске пакета;
- Пакеты, от которых зависит данный пакет, должны либо быть доступны в `тестируемом` выпуске, либо должны быть приняты в `тестируемый` выпуск одновременно с этим пакетом (эти пакеты будут приняты в случае, если они удовлетворяют всем необходимым критериям);
- Фаза проекта. Напр., автоматические переходы отключаются во время *заморозки* `тестируемого` выпуска.

Чтобы узнать, продвигается пакет к переходу в `тестируемый` выпуск или нет, см. вывод сценария `тестируемого` выпуска на веб-странице тестируемого выпуска, либо используйте программу `grep-excuses`, которая является частью пакета `devscripts`. Данная утилита легко может использоваться в crontab 5 для того, чтобы у вас всегда имелась информация о продвижении ваших пакетов в `тестируемый` выпуск.

Файл `update_excuses` не всегда содержит точную причину того, почему пакет был отклонён; вам может потребоваться выяснить это самостоятельно, проверяя, что может сломаться из-за добавления вашего пакета. Веб-страница тестируемого выпуска содержит немного больше информации об обычных проблемах, которые могут вызывать подобные затруднения.

Бывает так, что некоторые пакет никогда не включаются в `тестируемый` выпуск из-за того, что набор взаимных зависимостей слишком сложен и не может быть разобран сценариями. Подробности см. ниже.

Some further dependency analysis is shown on https://release.debian.org/migration/ — but be warned: this page also shows build dependencies that are not considered by britney.

5.16.2.1 Устаревание

For the `testing` migration script, outdated means: There are different versions in `unstable` for the release architectures (except for the architectures in outofsync_arches; outofsync_arches is a list of architectures that don't keep up (in `britney.py`), but currently, it's empty). Outdated has nothing whatsoever to do with the architectures this package has in `testing`.

Рассмотрим следующий пример:

	alpha	arm
testing	1	-
unstable	1	2

The package is out of date on `alpha` in `unstable`, and will not go to `testing`. Removing the package would not help at all; the package is still out of date on `alpha`, and will not propagate to `testing`.

Тем не менее, если сопровождающий ftp удаляет пакет в `нестабильном` выпуске (здесь — на архитектуре `arm`):

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

В этом случае пакет считается обновлённым на всех выпускаемых архитектурах в `нестабильном` выпуске (дополнительная архитектура `hurd-i386` не имеет значения, поскольку она не является выпускаемой архитектурой).

Время от времени возникает вопрос о том, можно ли разрешить переход пакетов, которые ещё не были собраны на всех архитектурах. Нет. Просто нет и всё. (За исключением случая, если вы сопровождаете glibc или что-то подобное.)

5.16.2.2 Удаление из тестируемого выпуска

Иногда какой-то пакет удаляется для того, чтобы был осуществлён переход другого пакета. Это происходит только для того, чтобы позволить *другому* пакету осуществить переход в случае, если он уже готов. Допустим, напр., что пакет `a` не может быть установлен вместе с новой версией пакета `b`; тогда пакет `a` может быть удалён для того, чтобы был осуществлён переход пакета `b`.

Of course, there is another reason to remove a package from `testing`: it's just too buggy (and having a single RC-bug is enough to be in this state).

Более того, если пакет был удалён из `нестабильного` выпуска, и ни один пакет в `тестируемом` выпуске не зависит от него, то этот пакет будет автоматически удалён.

5.16.2.3 Круговые зависимости

Ситуация, которая не может быть правильно обработана britney, заключает в том, что пакет `a` зависит от новой версии пакета `b`, и наоборот.

Пример:

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

Ни пакет `a`, ни пакет `b` не рассматриваются для обновления.

В настоящее время такая ситуация требует вмешательства выпускающей команды. Свяжитесь с ними, отправив сообщение по адресу `debian-release@lists.debian.org`, если такая ситуация возникла для одного из ваших пакетов.

5.16.2.4 Влияние пакета в тестируемом выпуске

Generally, there is nothing that the status of a package in `testing` means for transition of the next version from `unstable` to `testing`, with two exceptions: If the RC-bugginess of the package goes down, it may go in even if it is still RC-buggy. The second exception is if the version of the package in `testing` is out of sync on the different arches: Then any arch might just upgrade to the version of the source package; however, this can happen only if the package was previously forced through, the arch is in outofsync_arches, or there was no binary package of that arch present in `unstable` at all during the `testing` migration.

Короче говоря, это означает следующее: единственный фактор, на который влияет нахождение пакета в `тестируемом` выпуске, состоит в том, что новая версия этого пакета может быть проще добавлена.

5.16.2.5 Подробности

Если вам интересны подробности, то britney работает следующим образом:

Просматриваются пакеты на предмет выявления того, являются они корректными кандидатами или нет. Это даёт нам список оснований для отказа в обновлении. Наиболее частыми основаниями того, почему пакет не рассматривается для обновления, являются то, что он слишком нов, имеет критичные для выпуска ошибки, устарел на каких-то архитектурах. Для этой части britney у управляющих выпуском имеются различные инструменты, называемые подсказками (см. ниже), которые используются для того, чтобы заставить britney рассмотреть данный пакет.

Далее начинается более сложная часть. Britney пытается обновить `тестируемый` выпуск путём установки корректных кандидатов. Для этого britney пытается добавить каждого корректного кандидата в тестируемый выпуск. Если число неустанавливаемых пакетов в `тестируемом` выпуске не увеличивается, то пакет принимается. С этой точки зрения принятый пакет считается частью `тестируемого` выпуска, такой частью, что все последующий проверки установок будут включать в себя этот пакет. Подсказки выпускающей команды обрабатываются после или до этой основной работы в зависимости от типа подсказок.

If you want to see more details, you can look it up on https://release.debian.org/britney/update_output/.

The hints are available via https://release.debian.org/britney/hints/, where you can find the description as well. With the hints, the Debian Release team can block or unblock packages, ease or force packages into `testing`, remove packages from `testing`, approve uploads to *Прямые обновления тестируемого выпуска* or override the urgency.

5.16.3 Прямые обновления тестируемого выпуска

`Тестируемый` выпуск получает пакеты из `нестабильного` выпуска в соответствии с описанными ранее правилами. Тем не менее, в некоторых случаях необходимо загружать пакеты, собранные только для `тестируемого` выпуска. Для этого вы можете использовать загрузку в `testing-proposed-updates`.

Keep in mind that packages uploaded there are not automatically processed; they have to go through the hands of the release manager. So you'd better have a good reason to upload there. In order to know what a good reason is in the release managers' eyes, you should read the instructions that they regularly give on `debian-devel-announce@lists.debian.org`.

You should not upload to `testing-proposed-updates` when you can update your packages through `unstable`. If you can't (for example because you have a newer development version in `unstable`), you may use this facility. Even if a package is frozen, updates through `unstable` are possible, if the upload via `unstable` does not pull in any new dependencies.

Номера версий обычно выбираются путём добавления `+debXuY`, где X представляет собой мажорный номер выпуска Debian, а Y является счётчиком, начинающимся с `1`. Напр., `1:2.4.3-4+deb13u1`.

Убедитесь, что в вашей загрузке вы ничего не пропустили из следующего списка:

- Убедитесь, что ваш пакет действительно должен пройти через `testing-proposed-updates`, и что он не может пройти через `нестабильный` выпуск;
- Убедитесь, что вы включили в пакет лишь минимальное число изменений;
- Убедитесь, что вы включили соответствующее объяснение в журнал изменений пакета;
- Убедитесь, что вы указали *Кодовые имена выпусков* тестируемого выпуска (напр., `forky`) в качестве целевого выпуска;
- Убедитесь, что вы собрали и протестировали ваш пакет в `тестируемом`, а не в `нестабильном` выпуске;
- Убедитесь, что номер версии пакета выше номера версии, входящей в `тестируемый` выпуск и в `testing-proposed-updates`, а также ниже, чем в `нестабильном` выпуске;
- Ask for authorization for uploading from the release managers.
- После загрузки и успешной сборки на всех платформах, свяжитесь с выпускающей командой по адресу `debian-release@lists.debian.org` и попросите их одобрить вашу загрузку.

5.16.4 Часто задаваемые вопросы

5.16.4.1 Что такое критичные для выпуска ошибки, как производится их подсчёт?

Все ошибки, имеющие высокую важность, по умолчанию считаются критичными для выпуска; в настоящее время это ошибки с важностью `critical`, `grave` и `serious`.

Предполагается, что такие ошибки влияют на шансы того, будет пакет выпущен в составе `стабильного` выпуска Debian или нет. В общем случае, если пакет имеет открытые сообщения о критичных для выпуска ошибках, он не попадёт в `тестируемый` выпуск и, соответственно, не будет выпущен в составе `стабильного` выпуска.

The `unstable` bug count comprises all release-critical bugs that are marked to apply to *package*/*version* combinations available in `unstable` for a release architecture. The `testing` bug count is defined analogously.

5.16.4.2 Как установка какого-то пакета в `тестируемый` выпуск может сломать другие пакеты?

Структура архивов выпусков такова, что они могут содержать только одну версию пакета; пакет определяется его именем. Поэтому, когда пакет с исходным кодом `acmefoo` устанавливается в `тестируемый` выпуск вместе с соответствующими двоичными пакетами `acme-foo-bin`, `acme-bar-bin`, `libacme-foo1` и `libacme-foo-dev`, старая версии пакетов удаляются.

However, the old version may have provided a binary package with an old soname of a library, such as `libacme-foo0`. Removing the old `acmefoo` will remove `libacme-foo0`, which will break any packages that depend on it.

Evidently, this mainly affects packages that provide changing sets of binary packages in different versions (in turn, mainly libraries). However, it will also affect packages upon which versioned dependencies have been declared of the ==, <=, or << varieties.

When the set of binary packages provided by a source package changes in this way, all the packages that depended on the old binaries will have to be updated to depend on the new binaries instead. Because installing such a source package into `testing` breaks all the packages that depended on it in `testing`, some care has to be taken now: all the depending packages must be updated and ready to be installed themselves so that they won't be broken, and, once everything is ready, manual intervention by the release manager or an assistant is normally required.

Если у вас имеются подобные проблемы со сложной группой пакетов, свяжитесь с `debian-devel@lists.debian.org` or `debian-release@lists.debian.org` for help.

5.17 The Stable backports archive

5.17.1 Основы

Once a package reaches the `testing` distribution, it is possible for anyone with upload rights in Debian (see below about this) to build and upload a backport of that package to `stable-backports`, to allow easy installation of the version from `testing` onto a system that is tracking the `stable` distribution.

One should not upload a version of a package to `stable-backports` until the matching version has already reached the `testing` archive.

5.17.2 Exception to the testing-first rule

The only exception to the above rule, is when there's an important security fix that deserves a quick upload: in such a case, there is no need to delay an upload of the security fix to the `stable-backports` archive. However, it is strongly advised that the package is first fixed in `unstable` before uploading a fix to the `stable-backports` archive.

5.17.3 Who can maintain packages in the stable-backports archive?

It is not necessarily up to the original package maintainer to maintain the `stable-backports` version of the package. Anyone can do it, and one doesn't even need approval from the original maintainer to do so. It is however good practice to first get in touch with the original maintainer of the package before attempting to start the maintenance of a package in `stable-backports`. The maintainer can, if they wish, decide to maintain the backport themselves, or help you doing so. It is not uncommon, for example, to apply a patch to the unstable version of a package, to facilitate its backporting.

5.17.4 When can one start uploading to stable-backports?

The new `stable-backports` is created before the freeze of the next `stable` suite. However, it is not allowed to upload there until the very end of the freeze cycle. The `stable-backports` archive is usually opened a few weeks before the final release of the next `stable` suite, but it doesn't make sense to upload until the release has actually happened.

5.17.5 How long must a package be maintained when uploaded to stable-backports?

The `stable-backports` archive is maintained for bugs and security issues during the whole life-cycle of the Debian `stable` suite. Therefore, an upload to `stable-backports`, implies a willingness to maintain the backported package for the duration of the `stable` suite, which can be expected to be about 3 years from its initial release.

The person uploading to backports is also supposed to maintain the backported packages for security during the lifetime of `stable`.

It is to be noted that the `stable-backports` isn't part of the LTS or ELTS effort. The `stable-backports` FTP masters will close the `stable-backports` repositories for uploads once `stable` reaches end-of-life (ie: when `stable` becomes maintained by the LTS team only). Therefore there won't be any maintenance of packages from `stable-backports` after the official end of life of the `stable` suite, as uploads will not be accepted.

5.17.6 How often shall one upload to stable-backports?

The packages in backports are supposed to follow the developments that are happening in Testing. Therefore, it is expected that any significant update in `testing` should trigger an upload into `stable-backports`, until the new `stable` is released. However, please do not backport minor version changes without user visible changes or bugfixes.

5.17.7 How can one learn more about backporting?

You can learn more about how to contribute directly on the backport web site.

It is also recommended to read the Frequently Asked Questions (FAQ).

Глава 6

Лучшие практики создания пакетов

Debian's quality is largely due to the Debian Policy, which defines explicit baseline requirements that all Debian packages must fulfill. Yet there is also a shared history of experience which goes beyond the Debian Policy, an accumulation of years of experience in packaging. Many very talented people have created great tools, tools which help you, the Debian maintainer, create and maintain excellent packages.

Эта глава содержит описание некоторых лучших практик для разработчиков Debian. Все рекомендации являются только рекомендациями, а не требованиями или правилами. Они представляют собой лишь субъективные подсказки, советы и указания, позаимствованные у других разработчиков Debian. Выбирайте то, что лучше всего подходит именно вам.

6.1 Лучшие практики для `debian/rules`

The following recommendations apply to the `debian/rules` file. Since `debian/rules` controls the build process and selects the files that go into the package (directly or indirectly), it's usually the file maintainers spend the most time on.

6.1.1 Сценарии-помощники

Основная причина использования сценариев-помощников в файле `debian/rules` заключается в том, что последние позволяют сопровождающим распространять общую логику на множество пакетов и использовать её. Рассмотрим для примера вопрос установки пунктов меню: вам необходимо поместить файл в `/usr/share/menu` (или, если это требуется, `/usr/lib/menu` для выполняемых двоичных файлов меню), а также добавить команды сценариям сопровождающего для регистрации и отмены регистрации этих пунктов меню. Поскольку это довольно частая задача, зачем каждому сопровождающему переписывать всё это самостоятельно, а иногда и с ошибками? Кроме того, допустим, что каталог меню изменился, в этом случае пришлось бы изменить каждый пакет.

Сценарии-помощники могут позаботиться об этих проблемах. Допустим, вы соблюдаете конвенции, ожидаемые сценарием-помощником, а последний заботится обо всех деталях. Если политика изменится, то это изменение может быть отражено в сценарии-помощнике; тогда пакеты нужно будет лишь собрать заново с новой версией помощника без каких-либо изменений вручную.

Обзор инструментов Debian для сопровождающего содержит описание различных помощников. Наиболее часто используемой и лучшей (по нашему мнению) системой-помощником является `debhelper`. Предшествующие системы-помощники, такие как `debmake`, были монолитны: вы не могли выбрать только ту часть помощника, которая кажется вам полезной, вам приходилось использовать этот помощник для всего сразу. Тем не менее, `debhelper` представляет собой ряд отдельных

небольших программ `dh_*`. Например, `dh_installman` устанавливает и сжимает страницы руководства, `dh_installmenu` устанавливает файлы меню и так далее. Таким образом, этот помощник предлагает достаточный уровень гибкости, позволяющий использовать небольшие сценарии-помощники там, где это подходит, а также команды, добавленные в файл `debian/rules` вручную.

You can get started with `debhelper` by reading debhelper(7) and looking at the examples that come with the package. `dh_make`, from the `dh-make` package (see *dh-make*), can be used to convert a vanilla source package to a `debhelper`ized package. This shortcut, though, should not convince you that you do not need to bother understanding the individual `dh_*` helpers. If you are going to use a helper, you do need to take the time to learn to use that helper, to learn its expectations and behavior.

6.1.2 Множественные двоичные пакеты

A single source package will often build several binary packages, either to provide several flavors of the same software (e.g., the `vim` source package) or to make several small packages instead of a big one (e.g., so the user can install only the subset needed, and thus save some disk space, see for example the `libxml2` source package).

Со вторым случае можно легко управиться при помощи настроек в файле `debian/rules`. Вам нужно лишь переместить соответствующие файлы из каталога сборки во временные деревья пакета. Вы можете сделать это при помощи команд `install` или `dh_install` из пакета `debhelper`. Не забудьте проверить разные изменения различных пакетов на предмет того, что вы правильно установили межпакетные зависимости в файле `debian/control`.

The first case is a bit more difficult since it involves multiple recompiles of the same software but with different configuration options. The `vim` source package is an example of how to manage this using a hand-crafted `debian/rules` file.

6.2 Лучшие практики для `debian/control`

Следующие практики относятся к файлу `debian/control`. Они дополняют Политику описаний пакетов.

Описание пакета, как оно определено в соответствующем поле файла `control`, содержит резюме пакета и длинное описание этого пакета. *Общие принципы описания пакетов* описывает общие принципы обеих этих частей описания пакета. Далее, *Резюме пакета или короткое описание* содержит принципы, относящиеся к резюме, а *Длинное описание* представляет собой принципы, относящиеся к описанию.

6.2.1 The package name

The package name:

- Must be consistent with widely established conventions, e.g.
  - C and C++ libraries are typically prefixed with lib
  - for many other languages, package names are prefixed or suffixed with the language name (the actual convention depends on the language).
- Should not be a common, unqualified word. In particular, words that represent a whole class of applications (e.g. reader, browser) should be reserved for virtual packages.
- Should ideally be at least 4 characters long, unless the upstream name is shorter than that *and* the project is already widely recognized by that name (e.g. fzf).

6.2.2 Общие принципы описания пакетов

Описание пакета должно быть написано для среднего пользователя, среднего человека, который будет использовать и извлекать пользу из данного пакета. Например, пакеты для разработки предназначены для разработчиков, и могут быть описаны техническим языком. Описания приложений

общего назначения, таких как текстовые редакторы, должны быть написаны для менее подкованного в техническом плане пользователя.

Наш обзор описаний пакетов привёл нас к заключению, что большая часть описаний пакетов являются техническими, то есть они не написаны так, чтобы быть понятными технически не подкованным пользователям. Если ваш пакет предназначен не только для технических пользователей, это является проблемой.

Как следует писать описания для технически не подкованных пользователей? Избегайте жаргона. Избегайте указания на другие приложения или наборы приложений, которые могут быть незнакомы пользователю — GNOME или KDE это нормально, так как пользователи вероятно знакомы с этими терминами, но термин GTK скорее всего им не знаком. Попытайтесь вообще не допускать какого-либо знания. Если вам нужно использовать технические термины, определите их.

Будьте объективны. Описания пакетов — это не то место, где следует пропагандировать свой пакет, не важно, насколько вы его любите. Помните, что читатель может и не заботиться о тех же вещах, что и вы.

Указания на названия любых других пакетов ПО, названия протоколов, стандартов или спецификаций должны иметь каноническую форму, если таковая существует. Например, используйте X Window System, X11 или X; но не X Windows, X-Windows или X Window. Используйте GTK, а не GTK+ или gtk. Используйте GNOME, а не Gnome. Используйте PostScript, а не Postscript или postscript.

Если у вас возникли проблемы с написанием описания, вы можете отправить его по адресу `debian-l10n-english@lists.debian.org` с запросом помощи.

6.2.3 Резюме пакета или короткое описание

Политика говорит, что строка резюме (короткое описание) должна быть точной, не должна повторять имя пакета, но также должна быть информативной.

Резюме представляет собой фразу, описывающую пакет, а не полное предложение, поэтому пунктуация, характерная для предложений, тут неуместна: дополнительные заглавные буквы или точка в конце (как знак полной остановки) не нужны. Также следует пропустить любой начальный неопределённый или определённый артикль — "a", "an" или "the". Таким образом, например:

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

Технически это именная фраза без артиклей в противоположность глагольной фразе. Хорошая эвристика состоит в том, что должно быть возможно подставить *имя* пакета вместо его *резюме* в следующую формулу:

Пакет *имя пакета* предоставляет {a,an,the,some} *резюме пакета*.

Наборы связанных пакетов могут использовать альтернативную схему, которая разделяет резюме на две части, на, во-первых, описание всего набора и, во-вторых, резюме роли конкретного пакета в этом наборе:

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

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

Эти резюме следуют изменённой формуле. В ней пакет "*имя пакета*" имеет резюме "*набор пакетов* (*роль*)" или "*набор пакетов* - *роль*", элементы резюме должны быть сформулированы так, чтобы они подходили под следующую формулу:

Пакет *имя пакета* предоставляет {a,an,the} *роль* для *набора пакетов*.

6.2.4 Длинное описание

Длинное описание является первичной информацией, доступной пользователю о пакете до того, как он его установит. Оно должно предоставлять всю информацию, которая необходима для того, чтобы пользователь мог решить устанавливать ему этот пакет или нет. Допускайте, что пользователь уже прочитал резюме пакета.

Длинное описание должно состоять из полных и законченных предложений.

Первый абзац длинного описания должен содержать ответ на следующие вопросы: кто делает этот пакет? какие задачи он помогает решить пользователю? Важно описать это без использования технических терминов, если, конечно, аудиторией данного пакета не являются с необходимостью технически подкованные пользователи.

Long descriptions of related packages, for example built from the same source, can share paragraphs in order to increase consistency and reduce the workload for translators, but you need at least one separate paragraph describing the package's specific role.

The following paragraphs should answer the following questions: Why do I as a user need this package? What other features does the package have? What outstanding features and deficiencies are there compared to other packages (e.g., if you need X, use Y instead)? Is this package related to other packages in some way that is not handled by the package manager (e.g., is this the client for the foo server)?

Будьте внимательны, избегайте орфографических и грамматических ошибок. Проверьте описание на ошибки. И `ispell`, и `aspell` имеют специальные режимы для проверки файлов `debian/control`:

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

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

Обычно пользователи ожидают найти в описании пакета ответы на следующие вопросы:

- Что этот пакет делает? Если он является дополнением для другого пакета, то в описании следует поместить краткое описание того пакета, дополнением которого является данный пакет.
- Почему я хочу получить этот пакет? Ответ на этот вопрос связан с ответом на предыдущий вопрос, но это разные вопросы (это клиент электронной почты; он классный, быстрый, работает с PGP и LDAP, а также с IMAP, он имеет X, Y и Z).
- Если этот пакет не должен быть установлен напрямую, но тянется за другим пакетом, это обстоятельство должно быть явно указано.
- Если пакет является `экспериментальным`, либо имеются другие причины, по которым этот пакет не следуем использовать, если имеются другие пакеты, которые следует использовать вместо данного, это также должно быть отдельно указано.
- How is this package different from the competition? Is it a better implementation? more features? different features? Why should I choose this package?

6.2.5 Домашняя страница основной ветки разработки

Мы рекомендуем вам добавить ссылку на домашнюю страницу пакета в поле `Homepage` раздела `Source` в файле `debian/control`. Добавление этой информации в само описание пакета считается устаревшей нормой.

6.2.6 Размещение системы контроля версий

В файле `debian/control` имеются дополнительные поля для указания размещения системы контроля версий.

6.2.6.1 Vcs-Browser

Value of this field should be a `https://` URL pointing to a web-browsable copy of the Version Control System repository used to maintain the given package, if available.

Подразумевается, что данная информация будет полезна для конечного пользователя, который захочет ознакомиться с последней работой, связанной с этим пакетом (напр., захочет найти заплату для исправления ошибки, обозначенной с системе отслеживания ошибок тегом `pending`).

6.2.6.2 Vcs-*

Value of this field should be a string identifying unequivocally the location of the Version Control System repository used to maintain the given package, if available. `*` identifies the Version Control System; currently the following systems are supported by the package tracking system: `arch`, `bzr` (Bazaar), `cvs`, `darcs`, `git`, `hg` (Mercurial), `mtn` (Monotone), `svn` (Subversion).

Подразумевается, что эта информация будет полезна пользователю, которых осведомлён о принципах работы данной системы управления версиями и желает собрать текущую версию пакета из исходного кода, размещённого там. Другие способы использования этой информации могут включать автоматическую сборку последней версии данного пакета из системы управления версиями. Для этого размещение, указанное в данном поле, не должно иметь привязку к версии и должно указывать на основную ветку (это касается систем управления версиями, поддерживающих эту возможность). Кроме того, указываемое размещение должно быть доступно конечному пользователю; для выполнения этого требования может потребоваться указание на анонимный доступ к репозиторию вместо указания версии, доступной через SSH.

In the following example, an instance of the field for a Git repository of the `vim` package is shown. Note how the URL is in the `https://` scheme (instead of `ssh://`). The use of the `Vcs-Browser` and `Homepage` fields described above is also shown.

```
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
```

Maintaining the packaging in a version control system, and setting a Vcs-* header is good practice and makes it easier for others to contribute changes.

Almost all packages in Debian that use a version control system use Git; if you create a new package, using Git is a good idea simply because it's the system that contributors will be familiar with.

DEP-14 defines a common layout for Debian packages.

6.3 Лучшие практики для `debian/changelog`

Следующие практики дополняют Политику о файлах изменений.

6.3.1 Написание полезных пунктов файла изменений

Записи в файле изменений ревизии пакета описывают изменения данной ревизии и только их. Сосредоточьтесь на описании существенных и заметных для пользователя изменений, которые были произведены с момента выпуска последней версии.

Обратите внимание на то, *что* было изменено — кто, как и когда обычно менее важно. Указав это, не забудьте вежливо указать людей, которые оказали заметную помощь в создании пакета (напр., тех, что выслал заплаты).

Нет необходимости указывать тривиальные и очевидные изменения. Также вы можете объединять несколько изменений в одну запись. С другой стороны, не будьте чересчур кратки, если внесли

крупное изменение. Будьте особенно ясны, если были произведены изменения, которые оказывают влияние на поведение программы. Для указания дальнейших объяснений используйте файл `README.Debian`.

Используйте общепринятый вариант английского языка, чтобы большинство читателей смогли вас понять. Избегайте аббревиатур, технического языка и жаргона, когда вы описываете изменения, закрывающие ошибки, особенно это касается ошибок, о которых сообщили пользователи, и которые не кажутся вам в каком-то особом смысле технически сложными. Будьте вежливы, не ругайтесь.

Иногда желательно указать в начале записи об изменениях имена файлов, которые были изменены. Тем не менее, нет необходимости явно указывать каждый изменённый файл, особенно если изменения были небольшими или повторяющимися. Используйте шаблоны с метасимволами.

Если вы ссылаетесь на ошибки, не предполагайте у пользователей какого-либо знания. Сообщите, в чём заключалась проблема, как она была исправлена, и добавьте строку closes: #nnnnn. Дополнительную информацию см. в *Когда ошибки исправляются путём новых загрузок*.

6.3.2 Selecting the upload urgency

The release team have indicated that they expect most uploads to `unstable` to use **urgency=medium**. That is, you should choose **urgency=medium** unless there is some particular reason for the upload to migrate to `testing` more quickly or slowly (see also *Обновления из нестабильного выпуска*). For example, you might select **urgency=low** if the changes since the last upload are large and might be disruptive in unanticipated ways.

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 Распространённые неправильные представления о записях об изменениях

Записи об изменениях **не** должны описывать общие проблемы при создании пакета (Эй, если вы ищете foo.conf, он находится в /etc/бла-бла-бла/.), поскольку предполагается, что администраторы и пользователи по меньшей мере отдалённо знакомы с тем, как такие вещи обычно улаживаются в системах Debian. Сообщите, тем не менее, о том, что вы изменили расположение файла настройки, если вы произвели такое изменение.

Единственными закрываемыми с помощью записей об изменениях ошибками должны быть те, которые фактически исправляются именно в этой ревизии пакета. Закрытие несвязанных ошибок в журнале изменений является плохой практикой. См. *Когда ошибки исправляются путём новых загрузок*.

Записи об изменениях **не** должны использоваться для случайных дискуссий с теми, кто сообщил об ошибках (Я не наблюдаю ошибки сегментирования, когда запускаю foo с опцией bar; вышлите дополнительную информацию), общих утверждений о жизни, Вселенной и вообще (извините, загрузка этой версии заняла так много времени, но у меня была простуда), или просьб о помощи (список ошибок этого пакета очень велик, пожалуйста, помогите мне с этим). Скорее всего всё это не будет замечено целевой аудиторией, но может надоедать тем людям, которые хотят читать информацию о фактических изменениях в пакете. Подробную информацию о том, как использовать систему отслеживания ошибок см. в *Ответ на ошибки*.

Традиционно ошибки, исправленные в загрузках, которые произведены не сопровождающими, упоминаются в первой записи об изменениях той загрузки, которая произведена собственно сопровождающим. Поскольку сейчас у нас имеется система отслеживания версий, достаточно сохранить записи об изменениях, сделанных несопровождающими, и просто упомянуть этот факт в вашей записи об изменениях.

6.3.4 Общие ошибки в записях об изменениях

Следующие примеры демонстрируют некоторых общие ошибки или примеры плохого стиля в записях об изменениях.

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

Очевидно, эта запись не сообщает читателям ничего полезного.

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

Что это была за заплата?

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

Что это была за переделка? Упоминание поздней ночи предполагается как сообщение о том, что этому коду не следует доверять?

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

Too many acronyms (what does "fw" mean, "firmware"?), and it's not overly clear what the glitch was actually about, or how it was fixed.

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

Во-первых, вовсе не нужно загружать пакет, чтобы сообщить эту информацию; вместо этого используйте систему отслеживания ошибок. Во-вторых, отсутствует объяснение того, почему данный отчет об ошибке ошибкой не является.

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

If for some reason you didn't mention the bug number in a previous changelog entry, there's no problem, just close the bug normally in the BTS. There's no need to touch the changelog file, presuming the description of the fix is already in (this applies to the fixes by the upstream authors/maintainers as well; you don't have to track bugs that they fixed ages ago in your changelog).

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

Где описание? Если вы не можете придумать описание, начните со вставки заголовка каждой отдельной ошибки.

6.3.5 Дополнение журналов файлами NEWS.Debian

Важные новости об изменениях в пакете можно также поместить в файл `NEWS.Debian`. Новости будут отображены такими инструментами как `apt-listchanges` до всех остальных журналов изменений. Это является предпочтительным способом сообщения пользователям о существенных изменениях в пакете. Это лучше, чем использование примечаний `debconf`, поскольку этот способ менее надоедлив и пользователь может вернуться и просмотреть файл `NEWS.Debian` после установки. И это лучше, чем указание крупных изменений в файле `README.Debian`, поскольку пользователь легко может не заметить вашего сообщения.

Формат этот файла такой же как и формат файла журнала изменений, но забудьте о звёздочках и описывайте каждую новость в отдельном полном параграфе, если это необходимо, вместо того, чтобы давать более краткие резюме, которые подходят для журнала изменений. Хорошо бы пропустить ваш файл через `dpkg-parsechangelog` для проверки форматирования, поскольку в отличии от журнала изменений он не будет проверен автоматически во время сборки. Ниже приведён пример настоящего файла `NEWS.Debian`:

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

```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
   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
```

Файл `NEWS.Debian` устанавливается как `/usr/share/doc/`*пакет*`/NEWS.Debian.gz`. Он сжимается, и он всегда имеет данное имя даже в родных пакетах Debian. Если вы используете `debhelper`, `dh_installchangelogs` установит файлы `debian/NEWS` за вас.

В отличии от файлов журналов вам не нужно обновлять файлы `NEWS.Debian` в каждом выпуске. Обновляйте их только если имеется что-то действительно заслуживающее упоминания, что должно быть известно пользователю. Если у вас нет никаких новостей, не нужно добавлять файл `NEWS.Debian` в ваш пакет. Отсутствие новостей — уже хорошие новости!

6.4 Best practices around security

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 has suggestions on how to build security hardened executables.

6.5 Лучшие практики для сценариев сопровождающих

Maintainer scripts include the files `debian/postinst`, `debian/preinst`, `debian/prerm` and `debian/postrm`. These scripts take care of any package installation or deinstallation setup that isn't handled merely by the creation or removal of files and directories. The following instructions supplement the Debian Policy.

6.5.1 Generic guidelines

Сценарии сопровождающего должны быть идемпотентны. Это означает, что вам следует убедиться, что не произойдёт ничего плохого в том случае, когда сценарий будет вызван дважды, хотя обычно он вызывается только один раз.

Стандартные ввод и вывод могут быть перенаправлены (напр., конвейерами) с целью записи журнала для отладки, поэтому вам не следует считать, что они представляют собой лишь tty.

Все подсказки и интерактивные настройки должны быть сведены к минимуму. Когда это необходимо, для предоставления интерфейса следует использовать пакет `debconf`. Помните, что подсказки в любом случае могут быть выведены лишь на этапе `настройки` сценария `postinst`.

Делайте сценарии сопровождающего насколько простыми, насколько это возможно. Мы предлагаем использовать чистые сценарии оболочки POSIX. Помните, если вам нужны какие-либо возможности bash, сценарий сопровождающего должен содержать строку, объявляющую использование bash. Оболочка POSIX или Bash предпочтительны по отношению к Perl, поскольку они позволяют `debhelper` легко добавлять что-то новое в эти сценарии.

Если вы изменяете ваши сценарии сопровождающего, проверьте удаление пакета, двойную установку и очистку. Убедитесь, что очищенный пакет убран полностью, то есть, очистка должна удалить любые напрямую или косвенно созданные любым сценарием сопровождающего файлы.

Если вам необходимо проверить существование команды, используйте что-то вроде этого:

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

You can use this function to search `$PATH` for a command name, passed as an argument. It returns true (zero) if the command was found, and false if not. This is really the best way, since `command -v` is a shell-builtin for many shells and is defined in POSIX.

Using `which` is an acceptable alternative, since it is from the required `debianutils` package.

6.5.2 Expected behavior of complex postinst scripts

`postinst` scripts must normally not fail, except for very rare cases. These are cases where postinst must succeed:

on fresh installs:

- the service configuration is straightforward and can be reasonably expected to work as-is in typical Debian setups
- the service has no external dependencies (e.g. a database which may not yet be configured, or unreachable at install time)

on upgrades:

- `postinst` can verify with high confidence (which may not always be feasible) that the service was running prior to the restart
- the service has no external dependencies or `postinst` can verify that they are functional
- the service configuration has not changed in backwards incompatible ways between the old and new package versions

6.6 Управление настройкой с помощью `debconf`

`Debconf` is a configuration management system that can be used by all the various packaging scripts (`postinst` mainly) to request feedback from the user concerning how to configure the package. Direct user interactions must now be avoided in favor of `debconf` interaction. This will enable non-interactive installations in the future.

Debconf is a great tool but it is often poorly used. Many common mistakes are listed in the debconf-devel(7) man page. It is something that you must read if you decide to use debconf. Also, we document some best practices here.

Эти принципы включают некоторые рекомендации по типографии и стилю письма, общие соображения об использовании debconf, а также более конкретные рекомендации для некоторых частей дистрибутива (например, системы установки).

6.6.1 Не злоупотребляйте debconf

Поскольку debconf возник в Debian, им часто злоупотребляют, и дистрибутив Debian получил изрядную долю критики за злоупотребление debconf необходимостью отвечать на большое количество вопросов для установки даже очень небольшого пакета.

Keep usage notes to where they belong: the `NEWS.Debian`, or `README.Debian` file. Only use notes for important notes that may directly affect the package usability. Remember that notes will always block the install until confirmed or bother the user by email.

Carefully choose the questions' priorities in maintainer scripts. See debconf-devel 7 for details about priorities. Most questions should use medium and low priorities.

6.6.2 Общие рекомендации для авторов и переводчиков

6.6.2.1 Пишите на правильном английском

Большинство сопровождающих Debian не являются носителями английского языка. Поэтому написание правильно сформулированных шаблонов может быть для них нелегким делом.

Используйте список рассылки debian-l10n-english@lists.debian.org (и злоупотребляйте им). Пусть ваши шаблоны будут вычитаны.

Плохо написанные шаблоны создают плохое представление о вашем пакете, вашей работе... и даже самом Debian.

Избегайте технического жаргона насколько это возможно. Если некоторые термины звучат знакомо для вам, они всё равно могут быть непонятны остальным. Если вы не можете избежать этих терминов, попытайтесь объяснить их (используйте подробное описание). Если вы пишете описание, пытайтесь балансировать на грани подробности и простоты.

6.6.2.2 Будьте добры к переводчикам

Debconf templates may be translated. Debconf, along with its sister package po-debconf, offers a simple framework for getting templates translated by translation teams or even individuals.

Используйте шаблоны на основе gettext. Установите po-debconf в вашу систему, используемую для разработки, и прочтите документацию этого пакета (man po-debconf является хорошим началом).

Avoid changing templates too often. Changing template text induces more work for translators, which will get their translation fuzzied. A fuzzy translation is a string for which the original changed since it was translated, therefore requiring some update by a translator to be usable. When changes are small enough, the original translation is kept in PO files but marked as fuzzy.

Если вы планируете изменить ваши оригинальные шаблоны, для связи с переводчиками используйте систему оповещения, предоставляемую пакетом po-debconf, а именно podebconf-report-po. Наиболее активные переводчики очень отзывчивы, и включение их работы вместе с вашими изменёнными шаблонами позволит избежать вам дополнительных загрузок. Если вы используете шаблоны на основе gettext, имя переводчика и его адрес электронной почты указаны в заголовках PO-файлов и будут использоваться командой podebconf-report-po.

Рекомендованное использование этой утилиты:

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

This command will first synchronize the PO and POT files in debian/po with the template files listed in debian/po/POTFILES.in. Then, it will send a call for new translations, in the debian-i18n@lists.debian.org mailing list. Finally, it will also send a call for translation updates to the language team (mentioned in the Language-Team field of each PO file) as well as the last translator (mentioned in Last-translator).

Всегда ценится точный срок, отведённый для работы над переводом, чтобы переводчики могли организовать свою работу. Помните, что некоторые команды переводчиков используют формализованный процесс перевода/проверки, и задержка менее 10 дней считается чрезвычайно короткой. Более короткая задержка слишком сильно давит на команды переводчиков и должна устанавливаться только для очень небольших изменений.

Если у вас возникли сомнения, вы также можете связаться с командой перевода данного языка (debian-l10n-xxxxx@lists.debian.org) или списком рассылки debian-i18n@lists.debian.org.

6.6.2.3 Отменяйте статус неясных строк когда исправляете опечатки или орфографию

Когда текст шаблона debconf был изменён, и вы **уверены**, что это изменение **не** влияет на переводы, будьте добры по отношению к переводчикам и *отмените статус неясных строк* для их переводов.

Если вы не сделаете этого, весь шаблон не будет переведён пока переводчик не вышлет вам обновление.

Чтобы *отменить статус неясных строк* для переводов, вы можете использовать msguntypot (часть пакета po4a).

1. Создание файлов POT и PO заново.

```
debconf-updatepo
```

2. Создание копии POT-файла.

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

3. Создание копии всех PO-файлов.

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

4. Изменение файлов шаблона debconf для исправления ошибок.
5. Создание файлов POT и PO заново (опять).

```
debconf-updatepo
```

   Сейчас исправление опечатки привело к тому, что все переводы были отмечены как неясные, и это несчастное изменение является единственным различием между файлами PO вашего основного каталога и сохранёнными во временном каталоге файлами. Вот как решить эту проблема.

6. Отбросьте неясный перевод, восстановите перевод из сохранённых во временном каталоге файлов.

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

7. Вручную слейте файлы PO с новым POT-файлов, но учтите бесполезные неясности.

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

8. Очистите.

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

6.6.2.4 Не допускайте ничего по поводу интерфейсов

Templates text should not make reference to widgets belonging to some debconf interfaces. Sentences like *If you answer Yes...* have no meaning for users of graphical interfaces that use checkboxes for boolean questions.

Строки шаблонов должны также избегать в своём описании упоминания значений по умолчанию. Во-первых, это излишне, так как значения видны пользователям. Также это излишне потому, что значения по умолчанию могут различаться в зависимости от выбора сопровождающего (например, когда база данных debconf была автоматизирована).

Вообще говоря, попытайтесь избежать указания на действия пользователя. Просто представляйте факты.

6.6.2.5 Не используйте первое лицо

You should avoid the use of first person (*I will do this...* or *We recommend...*). The computer is not a person and the Debconf templates do not speak for the Debian developers. You should use neutral construction. Those of you who already wrote scientific publications, just write your templates like you would write a scientific paper. However, try using the active voice if still possible, like *Enable this if ...* instead of *This can be enabled if....*

6.6.2.6 Будьте нейтральны в гендерном отношении

As a way of showing our commitment to our diversity statement, please use gender-neutral constructions in your writing. This means avoiding pronouns like he/she when referring to a role (like "maintainer") whose gender is unknown. Instead, you should use the plural form (singular they).

6.6.3 Определение полей шаблонов

This part gives some information which is mostly taken from the debconf-devel(7) manual page.

6.6.3.1 Тип

string

Создаёт поле свободного ввода, в которое пользователь может ввести любую строку.

password

Запрашивает у пользователя пароль. Используйте с осторожностью; помните, что пароль, вводимый пользователем, будет записан в базу данных debconf. Вероятно, вам следует как можно скорее очистить это значение в базе данных.

boolean

Выбор истинно/ложно. Помните: истинно/ложно, **а не да/нет**...

select

Выбор одного значения из нескольких предложенных. Выбор должен быть определён в поле с именем 'Choices'. Разделите возможные значения запятыми и пробелами, например так: `Choices: yes, no, maybe`.

Если варианты выбора представляют собой переводимые строки, поле 'Choices' может быть отмечено как переводимое так: `__Choices`. Двойное подчёркивание выделит каждый вариант выбора в отдельную строку.

Также система `po-debconf` предлагает интересные возможности для отметки только **некоторых** вариантов выбора в качестве переводимых. Пример:

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

В этом примере только строка 'Other' является переводимой, остальные же представляют собой акронимы, которые не нужно переводить. Этот пример разрешает включение в файлы PO и POT только варианта 'Other'.

The debconf templates flag system offers many such possibilities. The po-debconf(7) manual page lists all these possibilities.

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

Этот тип данных не является вопросом самим по себе, а представляет собой примечание, которое может быть показано пользователю. Он должен использоваться только для важных примечаний, которые пользователи действительно должны заметить, поскольку debconf приложит максимум усилий для гарантии того, что пользователь заметит это примечание; установка будет остановлена до нажатия клавиши, в некоторых случаях пользователю будет даже выслано это примечание по почте.

text

Данный тип считается устаревшим: не используйте его.

error

This type is designed to handle error messages. It is mostly similar to the note type. Front ends may present it differently (for instance, the dialog front end of cdebconf draws a red screen instead of the usual blue one).

Рекомендуется использовать этот тип для любых сообщений, не которые необходимо обратить внимание пользователя для внесения исправлений любого вида.

6.6.3.2 Описание: краткое и расширенное описания

Описания шаблонов имеют две части: краткую и расширенную. Краткое описание даётся в строке Description: данного шаблона.

Краткое описание должно быть коротким (50 символов или около того), чтобы оно могло быть обработано большинством интерфейсов debconf. Краткое описание также помогает переводчикам, поскольку переводы обычно длиннее оригиналов.

The short description should be able to stand on its own. Some interfaces do not show the long description by default, or only if the user explicitly asks for it or even do not show it at all. Avoid things like: "What do you want to do?"

Краткое описание не обязательно должно быть полным предложением. Это лишь часть, она должна быть краткой и эффективной рекомендацией.

Расширенное описание не должно слово в слово повторять краткое описание. Если вы не можете придумать длинное описание, то для начала подумайте ещё. Напишите в debian-devel. Попросите помощи. Прослушайте курс письма! Расширенное описание очень важно. Если в конце концов вы всё еще не можете ничего придумать, оставьте его пустым.

Расширенное описание должно состоять из полных предложений. Параграфы должны быть короткими и удобочитаемыми. Не смешивайте две идеи в одном параграфе, лучше разделите их на два.

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.

Расширенное описание **никогда** не должно включать в себя вопрос.

Для ознакомления с конкретными правилами, зависящими от типа шаблона (string, boolean и т. д.), прочтите нижеследующий текст.

6.6.3.3 Choices

This field should be used for select and multiselect types. It contains the possible choices that will be presented to users. These choices should be separated by commas.

6.6.3.4 Default

Это поле опционально. Оно содержит ответ по умолчанию для шаблонов string, select и multiselect. Для шаблонов multiselect это поле может содержать список вариантов выбора, разделённых запятыми.

6.6.4 Template fields specific style guide

6.6.4.1 Тип поля

Без конкретного указания за исключением следующего: используйте соответствующий тип из предыдущего раздела.

6.6.4.2 Поле Description

Ниже приведены конкретные инструкции для правильного написания Description (краткого и расширенного описаний) в зависимости от типа шаблона.

Шаблоны string/password

- Краткое описание является приглашением, а **не** заголовком. Избегайте приглашений в виде вопросов (IP Address?) в пользу открытых приглашений (IP address:). Рекомендуется использовать двоеточие.
- Расширенное описание дополняет краткое описание. В расширенной части объясните пользователю, что спрашивается, а не задавайте ему тот же вопрос при помощи более длинных слов. Используйте полные предложения. Краткие стиль письма крайне не рекомендуется.

Шаблоны boolean

- The short description should be phrased in the form of a question, which should be kept short and should generally end with a question mark. Terse writing style is permitted and even encouraged if the question is rather long (remember that translations are often longer than original versions).
- Опять же, избегайте обращения к конкретным виджетам интерфейса. Частой ошибкой таких шаблонов являются ответы на конструкции Да-типа.

select/multiselect

- The short description is a prompt and **not** a title. Do **not** use useless "Please choose..." constructions. Users are clever enough to figure out they have to choose something... :)
- Расширенное описание должно заканчивать краткое описание. Оно может отсылать к доступным вариантам выбора. Также оно может содержать напоминание о том, что пользователь может выбрать более одного варианта из доступных, если выбран шаблон multiselect (хотя интерфейс обычно делает это очевидными).

Примечания

- Краткое описание должно рассматриваться как **заголовок**.
- Расширенное описание представляет собой то, что будет отображено в качестве более подробного объяснения данного примечания. Фразы, подробный стиль письма.
- **Do not abuse debconf.** Notes are the most common way to abuse debconf. As written in the debconf-devel(7) manual page: it's best to use them only for warning about very serious problems. The `NEWS.Debian` or `README.Debian` files are the appropriate location for a lot of notes. If, by reading this, you consider converting your Note type templates to entries in `NEWS.Debian` or `README.Debian`, please consider keeping existing translations for the future.

6.6.4.3 Поле Choices

Если скорее всего поле Choices будет часто меняться, попробуйте использовать трюк __Choices. Так каждый индивидуальный выбор будет выделен в отдельную строку, что поможет переводчикам делать их работу.

6.6.4.4 Поле Default

If the default value for a select template is likely to vary depending on the user language (for instance, if the choice is a language choice), please use the _Default trick, documented in po-debconf 7.

This special field allows translators to put the most appropriate choice according to their own language. It will become the default choice when their language is used while your own mentioned Default Choice will be used when using English.

Do not use an empty default field. If you don't want to use default values, do not use Default at all.

If you use po-debconf (and you **should**; see *Будьте добры к переводчикам*), consider making this field translatable, if you think it may be translated.

Пример, взятый из шаблонов пакета 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:
```

Note the use of brackets, which allow internal comments in debconf fields. Also note the use of comments, which will show up in files the translators will work with.

The comments are needed as the _Default trick is a bit confusing: the translators may put in their own choice.

6.7 Интернационализация

This section contains global information for developers to make translators' lives easier. More information for translators and developers interested in internationalization are available in the Internationalisation and localisation in Debian documentation.

6.7.1 Обработка переводов debconf

Подобно тем, кто занимается переносами, перед переводчиками стоит сложная задача. Они работают над многими пакетами и должны сотрудничать с большим количеством разных сопровождающих. Более того, по большей части они не являются носителями английского языка, поэтому вам следует быть с ними особенно вежливыми.

The goal of `debconf` was to make package configuration easier for maintainers and for users. Originally, translation of debconf templates was handled with `debconf-mergetemplate`. However, that technique is now deprecated; the best way to accomplish `debconf` internationalization is by using the `po-debconf` package. This method is easier both for maintainer and translators; transition scripts are provided.

Using `po-debconf`, the translation is stored in `.po` files (drawing from `gettext` translation techniques). Special template files contain the original messages and mark which fields are translatable. When you change the value of a translatable field, by calling `debconf-updatepo`, the translation is marked as needing attention from the translators. Then, at build time, the `dh_installdebconf` program takes care of all the needed magic to add the template along with the up-to-date translations into the binary packages. Refer to the po-debconf(7) manual page for details.

6.7.2 Интернационализованная документация

Интернационализация документации очень важна для пользователей, но требует большого труда. Не существует способа сделать так, чтобы не нужно было делать эту работу, но вы можете упростить её для переводчиков.

If you maintain documentation of any size, it is easier for translators if they have access to a source control system. That lets translators see the differences between two versions of the documentation, so, for instance, they can see what needs to be retranslated. It is recommended that the translated documentation maintain a note about what source control revision the translation is based on. An

interesting system is provided by doc-check in the `debian-installer` package, which shows an overview of the translation status for any given language, using structured comments for the current revision of the file to be translated and, for a translated file, the revision of the original file the translation is based on. You might wish to adapt and provide that in your VCS area.

If you maintain XML or SGML documentation, we suggest that you isolate any language-independent information and define those as entities in a separate file that is included by all the different translations. This makes it much easier, for instance, to keep URLs up to date across multiple files.

Some tools (e.g. `po4a`, `poxml`, or the `translate-toolkit`) are specialized in extracting the translatable material from different formats. They produce PO files, a format quite common to translators, which permits seeing what needs to be re-translated when the translated document is updated.

6.8 Best practices for `debian/patches`

Debian packages might suffer from bugs in the upstream code that you need to deal with. In the source format "3.0 (quilt)" patches are stored in `debian/patches/` and automatically applied as listed in `debian/patches/series` when the source package is unpacked.

Patches should be documented following DEP-3.

Several tools exist to automate managing the patches. If you manage a source package outside of any Git repository, then your best option is likely `quilt`. Otherwise, you should consider to rely on Git's built-in features or on on the git packaging helper that you use (if any). In particular, for packages using `git-buildpackage`, you should use the `gbp pq` commands to manage the contents of the `debian/patches/` directory.

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.

If the upstream fix is spread across multiple commits but makes sense to apply (and drop) in Debian as a single patch, one could use a command such as `git format-patch --stdout abc123..def456 > debian/patches/...` and append the `Bug` field only in the commit message of the first commit in the patch.

If one appends *.patch* to the url of a GitHub commit or Pull Request or GitLab commit or Merge Request, the resulting patch file is using this same format (as if it were generated by `git format-patch`).

Remember to always append a `Bug` header to the patch description so that a reader can follow the link to see where the bug was reported or patch submitted. If the purpose of the patch is to specifically divert from upstream permanently, append the header *Forwarded: not-needed* to the end of the description.

6.9 Общие ситуации при создании пакетов

6.9.1 Пакеты, использующие `autoconf/automake`

Поддержка файлов `autoconf` `config.sub` и `config.guess` в актуальном состоянии является критической задачей для тех, кто занимается переносом, особенно на более волатильные архитектуры. Некоторые очень хорошие практики по созданию пакетов для любого пакета, использующего `autoconf` и/или `automake` были синтезированы в `/usr/share/doc/autotools-dev/README.Debian.gz` из пакета `autotools-dev` package. Настойчиво рекомендуем прочитать этот файл и следовать приведённым в нём рекомендациям.

6.9.2 Библиотеки

По ряду причина для библиотек всегда трудно создавать пакеты. Политика налагает множество ограничений для облегчения из сопровождения и для того, чтобы гарантировать, что обновления будут настолько просты, насколько это возможно, когда выходит новая версия из основной ветки разработки. Поломка в библиотеке может вызвать поломку множества зависимых пакетов.

Good practices for library packaging have been grouped in the library packaging guide.

6.9.3 Документация

Обязательно пройдите по ссылке Политика документации.

Если ваш пакет содержит документацию, собираемую из XML или SGML, рекомендуем вам не добавлять исходный код в форматах XML или SGML в двоичный пакет (-ы). Если пользователи захотят получить исходный код документации, им следует загрузить пакет с исходным кодом.

Политика определяет, что документация должна поставляться в формате HTML. Мы также рекомендуем поставлять документацию в формате PDF и в виде обычного текста, если это удобно, и если возможно обеспечить вывод достаточного качества. Тем не менее, предоставление документации, исходным форматом которой является HTML, в виде обычного текста не является подходящим.

Крупные руководства должны при установке регистрироваться в `doc-base`. Дополнительную информацию см. в документации пакета `doc-base`.

Политика Debian (раздел 12.1) указывает, что справочные страницы должны поставляться со всякой программой, утилитой и функцией, и предлагает поставку справочных страниц для других объектов, таких как файлы настройки. Если та работа, для которой вы создаёте пакеты, не имеет таких справочных страниц, постарайтесь написать их самостоятельно для включения в ваш пакет и отправки в основную ветку разработки.

The manpages do not need to be written directly in the troff format. Popular source formats are DocBook, POD and reST, which can be converted using `xsltproc`, `pod2man` and `rst2man` respectively. To a lesser extent, the `help2man` program can also be used to write a stub.

6.9.4 Конкретные типы пакетов

Некоторые конкретные типы пакетов имеют свои специальные подполитики и соответствующие правила и практики создания пакетов:

- Perl related packages have a Perl policy; some examples of packages following that policy are `libdbd-pg-perl` (binary perl module) or `libmldbm-perl` (arch independent perl module).
- Python related packages have their Python policy; see `/usr/share/doc/python3/python-policy.txt.gz` in the `python3` package.
- Связанные с Emacs пакеты имеют Политику Emacs.
- Связанные с Java пакеты имеют свою Политику Java.
- OCaml related packages have their own policy, found in `/usr/share/doc/ocaml/ocaml_packaging_policy.gz` from the `ocaml` package. A good example is the `camlzip` source package.
- Пакеты, предоставляющие XML или SGML DTD должны соответствовать рекомендациям из пакета `sgml-base-doc`.
- Rust packaging is described in the Debian Rust Team Book;.
- Packages providing services ("daemons") should be functional on a fresh install, to the extent that that is possible without compromising security (e.g. a web server should by default be up and running and serve a dummy page, but must otherwise not allow unauthenticated sensitive operations; consider whether to serve only on the localhost network interface, by default).
- Web application packages should aim to have their dependencies (including javascript) packaged separately, and should carry out whatever setup is necessary for basic and secure functionality out of the box (e.g. create a database, ship configs with reasonable defaults, install files in appropriate location with appropriate permissions, etc). For examples, look at how existing web applications are packaged, e.g. dfsg-new-queue for Go, gitlab for ruby on rails, node-shiny-server for NPM. diaspora-installer is a dummy package which downloads diaspora (also pulling in runtime dependencies as rubygems) and configures it to use PostgreSQL and Nginx.

6.9.5 Независящие от архитектуры данные

Добавление большого количества независящих от архитектуры данных в пакеты с программами не является в каком бы то ни было смысле необычным явлением при создании пакетов. Например, аудио файлы, наборы иконок, шаблоны обоев рабочего стола или другие графические файлы. Если размер этих данных незначителен по сравнению с размером остального пакета, возможно лучшим решением будет сохранение всех данных в одном пакете.

However, if the size of the data is considerable, consider splitting it out into a separate, architecture-independent package (`_all.deb`). By doing this, you avoid needless duplication of the same data into ten or more .debs, one per each architecture. While this adds some extra overhead into the `Packages` files, it saves a lot of disk space on Debian mirrors. Separating out architecture-independent data also reduces processing time of `lintian` (see *Инструменты для проверки пакетов на предмет ошибок и соответствия стандартам*) when run over the entire Debian archive.

6.9.6 Необходимость в конкретной локали во время сборки

Если вам требуется определённая локаль во время сборки, вы можете создать временный файл с помощью следующих команд:

Если вы установите `LOCPATH` значение, эквивалентное `/usr/lib/locale`, а значение `LC_ALL` будет имя той локали, которую вы создаёте, у вас будет то, что вам нужно при работе из-за учётной записи суперпользователя. Что-то вроде этого:

```
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 Делаем так, чтобы deborphan определят переходные пакеты

deborphan — программам, помогающая пользователям определить то, какие пакеты могут быть безопасно удалены из системы, т. е. те, от которых не зависит более ни один пакет. По умолчанию поиск осуществляется только в разделах libs и oldlibs для обнаружения неиспользуемых библиотек. Но когда вы передали этой программе необходимых аргумент, она пытается найти все бесполезные пакеты.

For example, with `--guess-dummy`, `deborphan` tries to search all transitional packages which were needed for upgrade but which can now be removed. For that, it currently looks for the string dummy or transitional in their short description, though it would be better to search for both strings, as there are some dummy or transitional packages, which have other purposes.

So, when you are creating such a package, please make sure to add `transitional dummy package` to the short description to make this explicit. If you are looking for examples, just run: `apt-cache search .|grep dummy` or `apt-cache search .|grep transitional`.

Also, it is recommended to adjust its section to `oldlibs` and its priority to `optional` in order to ease `deborphan`'s job.

6.9.8 Лучшие практики для файлов `.orig.tar.{gz,bz2,xz}`

Существует два вида оригинальных tar-архивов с исходным кодом: чистый исходный код и запакованный заново исходный код основной ветки разработки.

6.9.8.1 Чистый исходный код

Определяющей характеристикой чистого tar-архива с исходным кодом является то, что файл `.orig.tar.{gz,bz2,xz}` побайтово идентичен tar-архиву, который официальной распространялся автором основной ветки разработки.[1] Это позволяет использовать контрольные суммы для простой проверки, что все изменения между версией Debian и версией основной ветки содержаться в db=iff-файле Debian. Кроме того, если оригинальный исходный код имеет большой размер, авторы основной ветки разработки и другие, кто уже имеют tar-архив из основной ветки, могут сохранить время, затрачиваемое на скачивание, в том случае, если они хотят подробно проверить ваш пакет.

There are no universally accepted guidelines that upstream authors follow regarding the directory structure inside their tarball, but `dpkg-source` is nevertheless able to deal with most upstream tarballs as pristine source. Its strategy is equivalent to the following:

1. Он распаковывает tar-архив в пустой временный каталог, выполняя

   ```
   zcat path/to/packagename_upstream-version.orig.tar.gz | tar xf -
   ```

2. Если после этого временный каталог не содержит ничего кроме одного каталога и не содержит других файлов, `dpkg-source` переименовывает этот каталог в *имяпакета*`-`*версия-основной-ветки*`(.orig)`. Имя каталога верхнего уровня в tar-архиве не имеет значения и просто забывается.

3. В противном случае, tar-архив основной ветки должен быть запакован без общего каталога верхнего уровня (позор автору основной ветки!). В этом случае `dpkg-source` переименовывает *самостоятельно* временный каталог в *имяпакета*`-`*версия-основной-ветки*`(.orig)`.

6.9.8.2 Повторная упаковка исходного кода основной ветки

Вы **должны** загружать пакеты с чистым исходным кодом, если это возможно, но имеются причины, по которым это может быть невозможно. Это имеет место, когда основная ветка вообще не поставляет исходный код в виде сжатого при помощи gzip tar-архива, либо если tar-архив основной ветки содержит не свободный в соответствии с критериями DFSG материал, который вам следует удалить до момента осуществления загрузки.

В этих случаях разработчик должен создать подходящий файл `.orig.tar.{gz,bz2,xz}` самостоятельно. Мы говорим о таком tar-архиве как о запакованном заново исходном коде основной ветки. Заметьте, что запакованный заново исходный код основной ветки отличается от родного пакета Debian. Запакованный заново исходный код содержит специфичные для Debian изменения в отдельных файлах `.diff.gz` или `.debian.tar.{gz,bz2,xz}` и всё ещё имеет номер версии, составленный из *версия-основной-ветки* и *версия-debian*.

Возможны такие случаи, когда желательно запаковать исходный код заново, даже несмотря на то, что основная ветка разработки поставляет `.tar.{gz,bz2,xz}`, который в принципе может использовать в его чистой форме. Наиболее очевидным случаем является ситуация, когда может быть достигнута *значительная* экономия места путём повторной упаковки tar-архива или путём удаления действительно ненужного хлама из архива основной ветки. Используйте это на своё усмотрение, но будьте готовы защищать своё решение, если вы запаковали заново исходный код, которые мог бы быть чистым.

Запакованный заново `.orig.tar.{gz,bz2,xz}`

1. **should** be documented in the resulting source package. Detailed information on how the repackaged source was obtained, and on how this can be reproduced should be provided in `debian/copyright`, ideally in a way that can be done automatically with uscan. If that really doesn't work, at least provide a `get-orig-source` target in your `debian/rules` file that repeats the process, even though that was actually deprecated in the 4.1.4 version of the Debian policy.

[1] We cannot prevent upstream authors from changing the tarball they distribute without also incrementing the version number, so there can be no guarantee that a pristine tarball is identical to what upstream *currently* distributing at any point in time. All that can be expected is that it is identical to something that upstream once *did* distribute. If a difference arises later (say, if upstream notices that they weren't using maximal compression in their original distribution and then re-`gzip` it), that's just too bad. Since there is no good way to upload a new `.orig.tar.{gz,bz2,xz}` for the same version, there is not even any point in treating this situation as a bug.

2. **не должен** содержать какие-либо файлы, которые получены не от авторов основной ветки разработки, или содержание которых было изменено вами.[2]

3. **should**, except where impossible for legal reasons, preserve the entire building and portability infrastructure provided by the upstream author. For example, it is not a sufficient reason for omitting a file that it is used only when building on MS-DOS. Similarly, a `Makefile` provided by upstream should not be omitted even if the first thing your `debian/rules` does is to overwrite it by running a configure script.

   (*Обоснование:* Обычно пользователи Debian, желающие собрать ПО для отличных от Debian платформ, загружают исходный код с зеркала Debian, а не пытаются найти каноничный выпуск основной ветки разработки).

4. **may** use *packagename-upstream-version*`+dfsg` (or any other suffix which is added to the tarball name) as the name of the top-level directory in its tarball. This makes it possible to distinguish pristine tarballs from repackaged ones.

5. **should** be compressed with `xz` (or `gzip` or `bzip2`) with maximal compression.

6.9.8.3 Изменение двоичных файлов

Sometimes it is necessary to change binary files contained in the original tarball, or to add binary files that are not in it. This is fully supported when using source packages in "3.0 (quilt)" format; see the dpkg-source(1) manual page for details. When using the older format "1.0", binary files can't be stored in the `.diff.gz` so you must store a `uuencode`d (or similar) version of the file(s) and decode it at build time in `debian/rules` (and move it in its official location).

6.9.9 Лучшие практики для отладочных пакетов

A debug package is a package that contains additional information that can be used by `gdb`. Since Debian binaries are stripped by default, debugging information, including function names and line numbers, is otherwise not available when running `gdb` on Debian binaries. Debug packages allow users who need this additional debugging information to install it without bloating a regular system with the information.

The debug packages contain separated debugging symbols that `gdb` can find and load on the fly when debugging a program or library. The convention in Debian is to keep these symbols in `/usr/lib/debug/`*path*, where *path* is the path to the executable or library. For example, debugging symbols for `/usr/bin/foo` go in `/usr/lib/debug/usr/bin/foo`, and debugging symbols for `/usr/lib/libfoo.so.1` go in `/usr/lib/debug/usr/lib/libfoo.so.1`.

6.9.9.1 Automatically generated debug packages

Debug symbol packages can be generated automatically for any binary package that contains executable binaries, and except for corner cases, it should not be necessary to use the old manually generated ones anymore. The package name for a automatic generated debug symbol package ends in `-dbgsym`.

The `dbgsym` packages are not installed into the regular archives, but in dedicated archives. That means, if you need the debug symbols for debugging, you need to add this archives to your apt configuration and then install the `dbgsym` package you are interested in. Please read https://wiki.debian.org/HowToGetABacktrace on how to do that.

6.9.9.2 Manual -dbg packages

Before the advent of the automatic `dbgsym` packages, debug packages needed to be manually generated. The name of a manual debug packages ends in `-dbg`. It is recommended to migrate such old legacy packages to the new `dbgsym` packages whenever possible. The procedure to convert your package is described in https://wiki.debian.org/AutomaticDebugPackages but the gist is to use the `--dbgsym-migration='pkgname-dbg (<< currentversion~)'` switch of the `dh_strip` command.

[2] As a special exception, if the omission of non-free files would lead to the source failing to build without assistance from the Debian diff, it might be appropriate to instead edit the files, omitting only the non-free parts of them, and/or explain the situation in a `README.source` file in the root of the source tree. But in that case please also urge the upstream author to make the non-free components easier to separate from the rest of the source.

However, sometimes it is not possible to convert to the new `dbgsym` packages, or you will encounter the old manual -dbg packages in the archives, so you might need to deal with them. It is not recommended to create manual -dbg packages for new packages, except if the automatic ones won't work for some reason.

One reason could be that debug packages contains an entire special debugging build of a library or other binary. However, usually separating debugging information from the already built binaries is sufficient and will also save space and build time.

This is the case, for example, for debugging symbols of Python extensions. For now the right way to package Python extension debug symbols is to use `-dbg` packages as described in https://wiki.debian.org/Python/DbgBuilds.

To create `-dbg` packages, the package maintainer has to explicitly specify them in `debian/control`.

The debugging symbols can be extracted from an object file using `objcopy --only-keep-debug`. Then the object file can be stripped, and `objcopy --add-gnu-debuglink` used to specify the path to the debugging symbol file. objcopy 1 explains in detail how this works.

Заметьте, отладочный пакет должен зависеть от пакета, для которого он предоставляет отладочные символы, и эта зависимость должна быть зависимостью от конкретной версии. Например:

```
Depends: libfoo (= ${binary:Version})
```

The `dh_strip` command in `debhelper` supports creating debug packages, and can take care of using `objcopy` to separate out the debugging symbols for you. If your package uses `debhelper/9.20151219` or newer, you don't need to do anything. `debhelper` will generate debug symbol packages (as `package-dbgsym`) for you with no additional changes to your source package.

6.9.10 Лучшие практики для метапакетов

Метапакет обычно является пустым пакетом, который облегчает установку связанного набора пакетов, которые со временем могут развиваться. Это достигается благодаря тому, что метапакет зависит от всех пакетов данного набора. Благодаря мощи APT, сопровождающий метапакета может изменять зависимости, и пользовательская система будет автоматически получать дополнительные пакеты. Кроме того, отброшенные пакеты, которые были установлены автоматически, будут помечены как кандидаты на удаление (и даже будут автоматически удалены с помощью `aptitude`). `gnome` и `linux-image-amd64` являются примерами метапакетов (собранных из пакетов с исходным кодом `meta-gnome2` и `linux-latest`).

The long description of the meta-package must clearly document its purpose so that the user knows what they will lose if they remove the package. Being explicit about the consequences is recommended. This is particularly important for meta-packages that are installed during initial installation and that have not been explicitly installed by the user. Those tend to be important to ensure smooth system upgrades and the user should be discouraged from uninstalling them to avoid potential breakages.

Глава 7

Помимо создания пакетов

Debian предполагает гораздо больше, чем простое создание пакетов ПО и сопровождение этих пакетов. Данная глава содержит информацию о способах, часто действительно критичных способах, участия в Debian помимо создания и сопровождения пакетов.

Будучи добровольной организацией, Debian полагается на благоразумие своих членов в их выборе того, над чем они хотят работать, а также в выборе наиболее критических задач, исходя из своего собственного времени.

7.1 Отправка отчётов об ошибках

Мы призываем вас сообщать об ошибках в пакетах Debian как только вы найдёте их. Фактически, разработчики Debian зачастую являются тестировщиками первой волны. Обнаружение ошибки и сообщение о ней другим разработчикам пакетов повышает качество Debian.

Прочтите инструкции по отправке отчётов об ошибках в систему отслеживания ошибок Debian.

Попытайтесь отправить сообщение об ошибке от учётной записи обычного пользователя, на которую вы получаете почту, чтобы люди могли с вами связаться, если им потребуется дополнительная информация об ошибке. Не отправляйте отчёты от учётной записи суперпользователя.

Вы можете использовать инструмент на подобие reportbug 1 для отправки сообщений об ошибках. Это может автоматизировать и в целом упростить процесс.

Убедитесь, что о данной ошибке в данном пакете ещё не сообщали. Всякий пакет имеет свой список ошибок, доступ к которому можно легко получить, по адресу `https://bugs.debian.org/`*`имя-пакета`*. Такие утилиты как querybts 1 также могут предоставить вам информацию (`reportbug` обычно вызывает команду `querybts` до отправки отчёта).

Попытайтесь направить ваше сообщение об ошибках туда, где они нужны. Когда, например, ваша ошибка касается пакета, который перезаписывает файлы из другого пакета, проверьте список ошибок, *оба* этих пакета нужны для того, чтобы избежать заполнения дублирующих отчётов об ошибках.

Для получения дополнительного кредита доверия вы можете просмотреть другие пакеты, сливая сообщения об ошибках, о которых было сообщено несколько раз, либо помечая ошибки как исправленные `fixed`, если они уже исправлены. Заметьте, что если вы не являетесь ни автором сообщения об ошибке, ни сопровождающим пакета, вам не следует фактически закрывать ошибку (если только вы не получили на это разрешение от сопровождающего).

Время от времени вам может захотеться проверить, что происходит с отчётами об ошибках, которые вы отправили. Используйте эту возможность, чтобы закрыть те ошибки, которые вы более не можете воспроизвести. Чтобы найти все ошибки, о которых вы отправляли отчёты, вам следует перейти на страницу `https://bugs.debian.org/from:` *ваш-адрес-электронной-почты*.

7.1.1 Отправка множества отчётов об ошибках за один раз (массовое заполнение отчётов об ошибках)

Сообщение о большом количестве ошибок, связанных с одной и той же проблемой в большом количестве различных пакетов — напр., более 10 — является плохой практикой. Предпримите все возможные шаги, чтобы не допустить отправки массы сообщений об ошибках. Например, если проверка данной проблемы может быть автоматизирована, добавьте новую проверку в `lintian`, так чтобы выводилось сообщение об ошибке или предупреждение.

Если вы сообщаете о более чем 10 ошибках на одну и ту же тему за раз, рекомендуется выслать сообщение на адрес `debian-devel@lists.debian.org` с описанием вашего намерения до того, как вы вышлете свой отчёт, укажите этот факт в теме сообщения. Это позволит другим разработчикам проверить, что ошибка действительно является проблемой. Кроме того, это позволит избежать ситуации, при которой несколько сопровождающих одновременно начинают заполнять один и тот же отчёт об ошибке.

Пожалуйста, используйте программы `dd-list` и, если это подходит, `whodepends` (из пакета `devscripts`) для создания списка всех подверженных данной ошибке пакетов и добавьте вывод этих программ в ваше сообщение по адресу `debian-devel@lists.debian.org`.

Заметьте, что когда вы высылаете сразу много сообщений об ошибках на одну и ту же тему, вам следует выслать отчёт об ошибке по адресу `maintonly@bugs.debian.org`, чтобы ваш отчёт об ошибке не был переслан в список рассылки об ошибках.

The program `mass-bug` (from the package `devscripts`) can be used to file bug reports against a list of packages.

7.1.1.1 Пользовательские метки

Возможно вам захочется использовать пользовательские метки системы отслеживания ошибок при отправке сообщений об ошибках в нескольких пакетах. Пользовательские метки (usertags) схожи с обычные метками, такими как 'patch' и 'wishlist', но отличаются от них в том, что они определяются пользователями и занимают пространство имён, уникальное для каждого отдельного пользователя. Это позволяет нескольким подмножествам разработчиков отмечать пользовательскими метками одну и ту же ошибку разными способами без возникновения конфликтов.

Чтобы добавить пользовательские метки при заполнении отчётов об ошибках, укажите псевдозаголовки `User` и `Usertags`:

```
To: submit@bugs.debian.org
Subject: title-of-bug

Package: pkgname
[ ... ]
User: email-addr
Usertags: tag-name [ tag-name ... ]

description-of-bug ...
```

Note that tags are separated by spaces and cannot contain underscores. If you are filing bugs for a particular group or team it is recommended that you set the `User` to an appropriate mailing list after describing your intention there.

Чтобы просмотреть сообщения об ошибках, помеченные конкретной пользовательской меткой, посетите страницу `https://bugs.debian.org/cgi-bin/pkgreport.cgi?users=`*адрес-электронной-почты*`&tag=`*имя-метки*.

7.2 Работа по контролю качества

7.2.1 Ежедневная работа

Даже несмотря на то, что у нас имеется выделенная группа людей, которые занимаются контролем качества (Quality Assurance), обязанности по контролю качества не резервируются исключительно за ними. Вы можете принять участие в этой работе, стараясь сделать так, чтобы в ваших пакетах было как можно меньше ошибок, и чтобы ваши пакеты проходили как можно большее количество проверок lintian (см. *lintian*). Если считаете, что это невозможно, вам, вероятно, следует отказаться (осиротить) некоторые из ваших пакетов (см. *Придание статуса осиротевшего пакета*). С другой стороны, вы можете попросить о помощи других людей, чтобы наверстать отставание по количеству исправленных ошибок (вы можете попросить о помощи в списках рассылки `debian-qa@lists.debian.org` или `debian-devel@lists.debian.org`). В то же время вы можете искать помощников сопровождающего (см. *Совместное сопровождение*).

7.2.2 Вечеринки по исправлению ошибок

From time to time the QA group organizes bug squashing parties to get rid of as many problems as possible. They are announced on `debian-devel-announce@lists.debian.org` and the announcement explains which area will be the focus of the party: usually they focus on release critical bugs but it may happen that they decide to help finish a major upgrade (like a new `perl` version that requires recompilation of all the binary modules).

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.

People participating in the party have special rules for NMU; they can NMU without prior notice if they upload their NMU to DELAYED/3-day at least. All other NMU rules apply as usual; they should send the patch of the NMU to the BTS (to one of the open bugs fixed by the NMU, or to a new bug, tagged fixed). They should also respect any particular wishes of the maintainer.

Если вы не чувствуете уверенности по поводу NMU, просто вышлите заплату в систему отслеживания ошибок. Это намного лучше, чем сломанная NMU.

7.3 Связь с другими сопровождающими

Во время срока вашей деятельности в Debian вам придётся связываться с другими сопровождающими по совершенно разным причинам. Возможно, вы захотите обсудить новый способ взаимодействия между наборами связных пакетов, или просто напомнить кому-то о том, что теперь доступна новая версия основной ветки разработки, и она вам нужна.

Поиск адреса электронной почты сопровождающего пакета может отвлекать. К счастью, существуют простые псевдонимы для электронной почты, *пакет*`@packages.debian.org`, которые предоставляют способ отправки сообщения сопровождающему, каким бы ни был его индивидуальный адрес электронной почты (или адреса). Замените *пакет* на имя пакета с исходным кодом или имя двоичного пакета.

Возможно, вы захотите связаться с теми, кто подписан на данный пакет с исходным кодом через систему отслеживания пакетов (*Система отслеживания пакетов Debian*). Вы можете сделать это, используя адрес электронной почты *пакет*`@packages.qa.debian.org`.

7.4 Работа с неактивными и/или недоступными сопровождающими

If you notice that a package is lacking maintenance, you should make sure that the maintainer is active and will continue to work on their packages. It is possible that they are not active anymore, but haven't

registered out of the system, so to speak. On the other hand, it is also possible that they just need a reminder.

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.

В качестве первого шага вежливо свяжитесь с сопровождающим, ждите его ответ разумное количество времени. Определить, какое количество времени является разумным, очень сложно, но важно принимать в расчёт то, что фактическая жизнь иногда может быть очень беспокойной. Например, можно отправить ещё одно напоминание через две недели.

A non-functional e-mail address is a violation of Debian Policy. If an e-mail "bounces", please file a bug against the package and submit this information to the MIA database.

Если сопровождающий не отвечает в течении четырёх недель (месяца), можно предположить, что он вероятнее всего более не ответит. Если это произошло, вам нужно расследовать дальше и попытаться собрать как можно больше полезной информации о данном сопровождающем. Это предполагает следующее:

- `Эшелон` информации доступен через базу данных LDAP разработчиков, в которой указывается, когда последний раз данный разработчик писал в список рассылки Debian. (База данных содержит информацию о почте по поводу загрузок, распространяемой через список рассылки `debian-devel-changes@lists.debian.org`.) Кроме того, проверьте, отмечен ли данный сопровождающий в этой базе данных как находящийся в отпуске.
- Число пакетов, за которые ответственен этот сопровождающий, и состояние этих пакетов. В частности, имеются ли какие-либо критичные для выпуска ошибки, которые остаются открытыми долгое время? Более того, как много вообще ошибок? Другой важной информацией является то, были ли эти пакеты загружены несопровождающим, и если да, то кем.
- Имеется ли какая-либо активность этого сопровождающего за пределами Debian? Например, он может написать что-то в список рассылки, не связанный с Debian, или в какую-нибудь новостную группу.

Некоторой проблемой являются пакеты, которые были поручены — их сопровождающий не является официальным разработчиком Debian. `Эшелон` информации не доступен для людей с поручителями, поэтому вам следует найти и связаться с разработчиком Debian, который фактически загрузил этот пакет. Учитывая, что они подписали пакет, они всё равно ответственны за загрузку и должны знать, что случилось с тем, за кого они поручались.

Также разрешается писать запрос в `debian-devel@lists.debian.org`, спрашивая о том, знает ли кто-нибудь о том, где находится пропавший сопровождающий. Пожалуйста, вышлите копию письма этому человеку.

Когда вы собрали всю эту информацию, вы можете связаться с `mia@qa.debian.org`. Люди, получающие почту с этого псевдонима, будут использовать предоставленную вами информацию для того, чтобы решить, как действовать дальше. Например, они могут осиротить один или все пакеты данного сопровождающего. Если пакет был загружен несопровождающим, они могут связаться с тем, что осуществил эту загрузку, до того, как осиротят пакет — возможно, человек, который осуществил эту загрузку, заинтересован в этом пакете.

Одно последнее слово: пожалуйста, будьте вежливы. Все мы являемся добровольцами, и мы не можем выделать всё наше время Debian. Кроме того, вам не известны обстоятельства, в которых находится этот человек. Возможно, он серьезно болен или даже мог умереть — вы не знаете, кто

получает его почту. Представьте, как будут чувствовать себя родные, если они прочтут сообщение, адресованное умершему, и это сообщение будет невежливым, злым или будет содержать обвинения!

On the other hand, although we are volunteers, a package maintainer has made a commitment and therefore has a responsibility to maintain the package. So you can stress the importance of the greater good — if a maintainer does not have the time or interest anymore, they should let go and give the package to someone with more time and/or interest.

If you are interested in working on the MIA team, please have a look at the `README` file in `/org/qa.debian.org/mia` on `qa.debian.org`, where the technical details and the MIA procedures are documented, and contact `mia@qa.debian.org`.

7.5 Dealing with repeatedly negligent Debian developers

Debian developers are expected to comply with a series of technical and non-technical policies (for example, respectively, the Debian Machine Usage Policy and the 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 Взаимодействие с будущими разработчиками Debian

Успех Debian зависит от его способности привлекать и сохранять новых и талантливых добровольцев. Если вы являетесь опытным разработчиком, мы рекомендуем вам принять участие в процессе привлечения новых разработчиков. Данный раздел описывает то, как помогать новым будущим разработчиками.

7.6.1 Поручение пакетов

Sponsoring a package means uploading a package for a maintainer who is not able to do it on their own. It's not a trivial matter; the sponsor must verify the packaging and ensure that it is of the high level of quality that Debian strives to have.

Разработчики Debian могут выступать поручителями для пакетов. Сопровождающие Debian не могут.

Процесс поручения пакета таков:

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 repositories and collaborative development platform*) where the package is maintained.
2. The sponsor downloads (or checks out) the source package.
3. Поручитель проверяет пакет с исходным кодом. Если будут найдены какие-либо проблемы, он информируем об этом сопровождающего и просит его предоставить исправленную версию (процесс начинается снова с шага 1).
4. Поручитель не нашёл каких-либо оставшихся проблем. Он собирает пакет, подписывает его и загружает пакет в Debian.

Before delving into the details of how to sponsor a package, you should ask yourself whether adding the proposed package is beneficial to Debian.

There's no simple rule to answer this question; it can depend on many factors: is the upstream codebase mature and not full of security holes? Are there pre-existing packages that can do the same task and how do they compare to this new package? Has the new package been requested by users and how large is the user base? How active are the upstream developers?

Кроме того, вам следует убедиться, что предполагаемый сопровождающий будет хорошим сопровождающим. Имеет ли он какой-либо опыт работы с другими пакетами? Если да, то хорошо ли он с ними работал (проверьте несколько ошибок)? Знаком ли он с пакетом и языком программирования, на котором написана программа? Имеет ли он требуемые для данного пакета навыки? Если нет, то способен ли он ими овладеть?

It's also a good idea to know where they stand with respect to Debian: do they agree with Debian's philosophy and do they intend to join Debian? Given how easy it is to become a Debian Member, you might want to only sponsor people who plan to join. That way you know from the start that you won't have to act as a sponsor indefinitely.

7.6.1.1 Поручительство нового пакета

New maintainers usually have certain difficulties creating Debian packages — this is quite understandable. They will make mistakes. That's why sponsoring a brand new package into Debian requires a thorough review of the Debian packaging. Sometimes several iterations will be needed until the package is good enough to be uploaded to Debian. Thus being a sponsor implies being a mentor.

Никогда не поручайтесь за новый пакет, если вы его не изучили. Изучение новых пакетов осуществляется сопровождающими FTP-архива, это гарантирует, что ПО действительно является свободным. Конечно, иногда они обнаруживают проблемы создания пакета, но вообще они не должны этим заниматься. Это ваша задача, вы должны гарантировать, что загружаемый пакет соответствует Критериям Debian по определению Свободного ПО, и что он создан качественно.

Сборка пакета и тестирование ПО являются частью исследования пакета, но этого не достаточно. Оставшаяся часть данного раздела содержит неполный список того, на что необходимо обратить внимание.[1]

- Убедитесь, что включённый в пакет tarball-архив, не отличается от архива, распространяемого автором основной ветки разработки (если архив с исходным кодом для Debian был создан заново, создайте изменённый tarball-архив самостоятельно).
- Run `lintian` (see *lintian*). It will catch many common problems. Be sure to verify that any `lintian` overrides set up by the maintainer are fully justified.
- 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.
- Соберите пакет с помощью `pbuilder` (или любого другого схожего инструмента, см. *pbuilder*) для того, чтобы гарантировать, что сборочные зависимости полны.
- Проверьте `debian/control`: подготовлен ли он в соответствии с лучшими практиками (см. *Лучшие практики для debian/control*)? Полны ли зависимости?
- Проверьте `debian/rules`: подготовлен ли он в соответствии с лучшими практиками (см. *Лучшие практики для debian/rules*)? Можно ли что-либо улучшить?
- Проверьте сценарии сопровождающего (`preinst`, `postinst`, `prerm`, `postrm`, `config`): будет ли `preinst`/`postrm` работать в том случае, если зависимости не установлены? Все ли сценарии идемпотентны (т. е., можете ли вы запустить их несколько раз без каких-либо последствий)?
- Проверьте изменения в файлах из основной ветки разработки (либо в `.diff.gz`, либо в `debian/patches/`, либо в самом встроенном tarball-архиве `debian` для двоичных файлов). Обоснованы ли эти изменения? Правильно ли они документированы (с DEP-3 для заплат)?
- Относительно каждого файла задайте себе вопрос о том, почему этот файл находится здесь, правильно достигнут желаемый результат. Следует ли сопровождающий лучшим практикам (см. *Лучшие практики создания пакетов*)?
- Build the packages, install them and try the software. Ensure that you can remove and purge the packages. Maybe test them with `piuparts`.

[1] You can find more checks in the wiki, where several developers share their own sponsorship checklists.

If the audit did not reveal any problems, you can build the package and upload it to Debian. Remember that even if you're not the maintainer, as a sponsor you are still responsible for what you upload to Debian. That's why you're encouraged to keep up with the package through *Система отслеживания пакетов Debian*.

Note that you should not need to modify the source package to put your name in the `changelog` or in the `control` file. The `Maintainer` field of the `control` file and the `changelog` should list the person who did the packaging, i.e. the sponsee. That way they will get all the BTS mail.

Instead, you should instruct `dpkg-buildpackage` to use your key for the signature. You do that with the `-k` option:

```
dpkg-buildpackage -kKEY-ID
```

Если вы используете `debuild` и `debsign`, вы даже можете создать постоянную настройку в файле `~/.devscripts`:

```
DEBSIGN_KEYID=KEY-ID
```

7.6.1.2 Поручение обновления существующего пакета

You will usually assume that the package has already gone through a full review. So instead of doing it again, you will carefully analyze the difference between the current version and the new version prepared by the maintainer. If you have not done the initial review yourself, you might still want to have a deeper look just in case the initial reviewer was sloppy.

To be able to analyze the difference, you need both versions. Download the current version of the source package (with `apt-get source`) and rebuild it (or download the current binary packages with `aptitude download`). Download the source package to sponsor (usually with `dget`).

Read the new changelog entry; it should tell you what to expect during the review. The main tool you will use is `debdiff` (provided by the `devscripts` package); you can run it with two source packages (`.dsc` files), or two binary packages, or two `.changes` files (it will then compare all the binary packages listed in the `.changes`).

Если вы сравниваете пакеты с исходным кодом (исключая файлы из основной ветки разработки в случае новой версии из основной ветки разработки, например, фильтруя вывод команды `debdiff` с помощью `filterdiff -i '*/debian/*'`), вам следует понять все изменения, которые вы видите, и они должны быть соответствующим образом документированы в журнале изменений Debian.

If everything is fine, build the package and compare the binary packages to verify that the changes on the source package have no unexpected consequences (some files dropped by mistake, missing dependencies, etc.).

You might want to check out the Package Tracking System (see *Система отслеживания пакетов Debian*) to verify if the maintainer has not missed something important. Maybe there are translation updates sitting in the BTS that could have been integrated. Maybe the package has been NMUed and the maintainer forgot to integrate the changes from the NMU into their package. Maybe there's a release critical bug that they have left unhandled and that's blocking migration to `testing`. If you find something that they could have done (better), it's time to tell them so that they can improve for next time, and so that they have a better understanding of their responsibilities.

Если вы не нашли серьёзных проблем, загрузите новую версию. В противном случае попросите сопровождающего предоставить вам исправленную версию.

7.6.2 Granting upload permissions to DMs

After a Debian Maintainer's key has been added to the debian-maintainers keyring, a Debian Developer may grant upload permissions to the DM for specific packages by uploading a signed dak command to ftp.upload.debian.org as described in the FTP-Master's announcement to debian-devel.

This process can be simplified with the help of the `dcut` command from the `dput-ng` package. Note that this does not work with the `dcut` command from the `dput` package!

For example:

```
dcut dm --uid 0xfedcba9876543210 --allow nano --deny bash
```

If the DM's key is not in the keyring package yet but in the DD's local keyring, use the `--force` option and the fingerprint, without spaces and, in this special case, without the 0x prefix and in all uppercase:

```
dcut --force dm --uid FEDCBA9876543210FEDCBA9876543210 --allow nano
```

7.6.3 Поддержка новых разработчиков

Смотрите страницу о поддержке будущих разработчиков на веб-сайте Debian.

7.6.4 Обработка заявок новых сопровождающих

Пожалуйста, посмотрите Перечень для менеджеров заявок на веб-сайте Debian.

Глава 8

Интернационализация и переводы

Debian поддерживает постоянно увеличивающееся число языков. Даже если вашим родным языком является английский, и вы не говорите ни на одном другом языке, как сопровождающий вы должны знать о процессе интернационализации (сокращённо i18n, так как в слове internationalization между 'i' и 'n' 18 букв). Следовательно, даже если вам вполне подходят программы с интерфейсом на английском языке, вам следует прочесть большую часть этой главы.

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 и i18n взаимосвязаны, но трудности, касающиеся каждого из этих процессов, весьма различны. Вообще-то не сложно сделать так, чтобы программа изменяла язык, на котором отображается текст, на основе пользовательских настроек, но перевести все эти сообщения весьма затратно по времени. С другой стороны, установить кодировку символов весьма легко, но изменить код так, чтобы можно было использовать несколько кодировок, довольно трудно.

Если оставить проблемы i18n, по поводу которых нет общего руководства, в стороне, можно видеть, что для l10n в Debian нет какой-либо центральной инфраструктуре, которая бы была сравнима с механизмом buildd для переноса на другие архитектуры. Поэтому большая часть работы выполняется вручную.

8.1 Как переводы обрабатываются в Debian

Обработка переводов текстов, содержащихся в пакете, всё ещё является задачей, которую приходится решать вручную, сам же этот процесс зависит от того, к какому виду относится текст, который вы желаете перевести.

For program messages, the gettext infrastructure is used most of the time. Often the translation is handled upstream within projects like the Free Translation Project, the GNOME Translation Project or the KDE Localization project. The only centralized resources within Debian are the Central Debian translation statistics, where you can find some statistics about the translation files found in the actual packages and download those files.

Package descriptions have translations since many years and Maintainers don't need to do anything special to support translated package descriptions; translators should use the Debian Description Translation Project (DDTP).

For `debconf` templates, maintainers should use the `po-debconf` package to ease the work of translators. Some statistics can be found on the Central Debian translation statistics site.

Для веб-страницы у каждой команды l10n имеется доступ к релевантной системе управления версиями, статистика доступна на сайте центральной статистики переводов Debian.

Для общей документации относительно Debian процесс перевода более или менее совпадает с процессом перевода веб-страниц (переводчики имеют доступ к системе управления версиями), но страницы со статистикой отсутствуют.

Another part of i18n work is package-specific documentation (man pages, info documents, other formats). At least the man page translations are po-based as most other things mentioned above.

8.2 ЧаВО по I18N и L10N для сопровождающих

Это список проблем i18n и l10n, с которыми могут столкнуться сопровождающие. Читая данный документ, помните, что в настоящее время никакого действительного согласия по этим вопросам в Debian нет, и что в документе содержатся только советы. Если у вас имеются более подходящие идеи для некоторой данной проблемы, либо если вы не согласны с чем-то, не стесняйтесь сообщить об этом, это поспособствует улучшению данного документа.

8.2.1 Как перевести некоторый данный текст

To translate package descriptions, you have nothing to do; the DDTP infrastructure will dispatch the material to translate to volunteers with no need for interaction on your part.

For all other material (`debconf` templates, gettext files, man pages, or other documentation), the best solution is to ask on debian-i18n for a translation in different languages. Some translation team members are subscribed to this list, and they will take care of the needed coordination, to get the material translated and reviewed. Once they are done, you will get your translated document from them in your mailbox or via a wishlist bugreport. It is also recommended, to use the `po-debconf` tools for i18n integration.

8.2.2 Как проверить перевод

Время от времени кто-то переводит различные тексты из ваших пакетов, эти люди будут просить вас включить их перевод в пакет. Это может быть проблемой, если вы не владеете данным языком. Лучше всего выслать данный документ в соответствующий список рассылки l10n и попросить о проверке. Когда это будет сделано, вы будете более уверены в качестве перевода, тогда уже можете спокойно добавлять его в ваш пакет.

8.2.3 Как обновить перевод

Если у вас имеются переводы какого-то текста, то всякий раз, когда вы обновляете оригинал, вам следует попросить предыдущего переводчика обновить его перевод для включения новых изменений. Помните, что эта задача требует некоторого времени; по меньшей мере, нужна одна неделя, чтобы обновить и проверить весь перевод.

Если переводчик не отвечает, вы можете попросить о помощи в соответствующем списке рассылки l10n. Если ничего не получилось, не забудьте добавить предупреждение в переведённый документ о том, что данный перевод устарел, и что читателю по возможности следует обратиться к оригинальному документу.

Избегайте полного удаления перевода даже в случае, если он сильно устарел. Для тех, что не говорит на английском устаревшая документация зачастую лучше, чем её полное отсутствие.

8.2.4 Как работать с отчётом об ошибке, касающемся перевода

Лучшим решением может состоять в следующем: отметьте сообщение об ошибке как пересланное в основную ветку разработки, затем перешлите её предыдущему переводчику и соответствующей команде локализации (используя соответствующий список рассылки debian-l10n-XXX).

8.3 ЧаВО по I18N и L10N для переводчиков

Читая данный документ, помните, что в Debian отсутствует какая-либо общая процедура, определяющая работу над переводами, в любом случае вам следует взаимодействовать с вашей командой и сопровождающим пакета.

8.3.1 Как помочь с переводом

Выберите, что бы вы хотели перевести, убедитесь, что никто не работает над переводом этого документа (используя ваш список рассылки debian-l10n-XXX), переведите, получите отзывы от других носителей языка в вашем списке рассылки l10n, предоставьте перевод сопровождающему пакета (см. следующий пункт).

8.3.2 Как предоставить перевод для добавления его в пакет

Убедитесь, что перевод верен (попросив проверить его в вашем списке рассылки l10n) до отправки его для включения в пакет. Это сохранит всем время и позволит избежать хаоса, когда в сообщениях об ошибках имеются несколько версий одного и того же документа.

The best solution is to file a regular bug containing the translation against the package. Make sure to use both the `patch` and `l10n` tags, and to not use a severity higher than 'wishlist', since the lack of translation never prevented a program from running.

8.4 Лучшие текущие практики, касающиеся локализации

- Как сопровождающему вам никогда не следует редактировать переводы (даже для изменения форматирования), не спросив об этом в соответствующем списке рассылки l10n. Вы рискуете, например, испортить кодировку файлов. Более того, то, что вы считаете ошибкой, может оказаться правильным (или даже необходимым) для данного языка.
- Как переводчику, если вы находите ошибку в оригинальном тексте, вам следует сообщить об этом. Переводчики часто являются более внимательными читателями некоторого данного текста, и если они не сообщают о найденных ошибках, то о них не сообщит никто.
- В любом случае помните, что главная проблема l10n состоит в том, что для локализации требуется взаимодействие нескольких людей, и что из-за непонимания очень и очень легко начать войну по поводу небольших проблем. Поэтому если у вас возникли проблемы с вашим собеседником, попросите о помощи в соответствующем списке рассылки l10n, в списке рассылки debian-i18n или даже в debian-devel (но остерегайтесь, обсуждения l10n в этом списке рассылки часто переходят в настоящие войны :)
- В любом случае взаимодействие может быть достигнут только благодаря **взаимному уважению**.

Глава 9

Обзор инструментов Debian для сопровождающего

Данный раздел содержит краткий обзор доступных сопровождающим инструментов. Приведённая ниже информация ни в коем мере не является полной или окончательной, она представляет собой лишь руководство по некоторым более популярным утилитам.

Инструменты сопровождающего Debian предназначены для облегчения работы разработчиком и освобождения их времени для решения критических задач. Как говорит Ларри Уолл, существует более одного способа сделать это.

Some people prefer to use high-level package maintenance tools and some do not. Debian is officially agnostic on this issue; any tool that gets the job done is fine. Therefore, this section is not meant to stipulate to anyone which tools they should use or how they should go about their duties of maintainership. Nor is it meant to endorse any particular tool to the exclusion of a competing tool.

Most of the descriptions of these packages come from the actual package descriptions themselves. Further information can be found in the package documentation itself. You can also see more info with the command `apt-cache show` *package-name*.

9.1 Базовые инструменты

Следующие инструменты весьма нужны любому сопровождающими.

9.1.1 dpkg-dev

`dpkg-dev` содержит инструменты (включая `dpkg-source`), необходимые для распаковки, сборки и загрузки пакетов Debian с исходным кодом. Эти утилиты предоставляют базовую, низкоуровневую функциональность, необходимую для создания пакетов и для манипуляции ими; как таковые, эти утилиты являются необходимыми любому сопровождающему Debian.

9.1.2 debconf

`debconf` предоставляет единообразный интерфейс для интерактивной настройки пакетов. Его пользовательских интерфейс независим, что позволяет конечным пользователям настраивать пакеты при помощи текстового интерфейса, интерфейса HTML, либо диалогового интерфейса. Новые варианты интерфейса могут быть добавлены в виде модулей.

Документацию для этого пакета вы можете найти в пакете `debconf-doc`.

Many feel that this system should be used for all packages that require interactive configuration; see *Управление настройкой с помощью debconf*. `debconf` is not currently required by Debian Policy, but that may change in the future.

9.1.3 fakeroot

`fakeroot` имитирует привилегии суперпользователя. Это позволяет вам собирать пакеты без привилегий суперпользователя (обычно пакеты хотят установить файлы, владельцем которых является суперпользователь). Если у вас установлен `fakeroot`, `dpkg-buildpackage` будет автоматически использовать его.

9.2 Инструменты для проверки пакетов на предмет ошибок и соответствия стандартам

According to the Free On-line Dictionary of Computing (FOLDOC), `lint` is: "A Unix C language processor which carries out more thorough checks on the code than is usual with C compilers." Package lint tools help package maintainers by automatically finding common problems and policy violations in their packages.

9.2.1 lintian

`lintian` анализирует пакеты Debian и выдаёт информацию об ошибках и нарушениях Политики. Он содержит автоматические проверки множества аспектов Политики Debian, а также некоторые проверки на наличие распространённых ошибок.

Периодически вам следует получить наиболее свежую версию `lintian` из `нестабильного` выпуска и проверять все ваши пакеты. Заметьте, что опция `-i` предоставляет подробное объяснение того, что означает каждая ошибка или предупреждение, на каком пункте Политики они основываются, а иногда и то, как вы можете исправить проблему.

Для получения дополнительной информации о том, как и когда использовать Lintian, обратитесь к *Тестирование пакета*.

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` contains a set of scripts that can automatically fix more than 80 common lintian issues in Debian packages.

It comes with a wrapper script that invokes the scripts, updates the changelog (if desired) and commits each change to version control.

9.2.3 piuparts

`piuparts` is the `.deb` package installation, upgrading, and removal testing tool.

`piuparts` tests that `.deb packages` handle installation, upgrading, and removal correctly. It does this by creating a minimal Debian installation in a chroot, and installing, upgrading, and removing packages in that environment, and comparing the state of the directory tree before and after. `piuparts` reports any files that have been added, removed, or modified during this process.

`piuparts` is meant as a quality assurance tool for people who create `.deb` packages to test them before they upload them to the Debian archive.

9.2.4 debdiff

`debdiff` (из пакета `devscripts`, *devscripts*) сравнивает списки файлов и управляющие файлы двух пакетов. Это простая проверка на наличие регрессий, поскольку она позволяет вам заметить, что число двоичных пакетов изменилось с момента последней загрузки, либо если что-то было изменено в управляющем файле. Конечно, некоторые изменения, о которых сообщает эта утилита, не являются чем-то плохим, но она может помочь вам предотвратить различные случайные проблемы.

Вы можете запустить её, указав два двоичных пакета:

```
debdiff package_1-1_arch.deb package_2-1_arch.deb
```

Или даже два файла changes:

```
debdiff package_1-1_arch.changes package_2-1_arch.changes
```

Дополнительную информацию см. в debdiff 1.

9.2.5 diffoscope

`diffoscope` provides in-depth comparison of files, archives, and directories.

`diffoscope` will try to get to the bottom of what makes files or directories different. It will recursively unpack archives of many kinds and transform various binary formats into more human readable form to compare them.

Originally developed to compare two `.deb` files or two `changes` files nowadays it can compare two tarballs, ISO images, or PDF just as easily and supports a huge variety of filetypes.

The differences can be shown in a text or HTML report or as JSON output.

9.2.6 duck

`duck`, the Debian Url ChecKer, processes several fields in the `debian/control`, `debian/upstream`, `debian/copyright`, `debian/patches/*` and `systemd.unit` files and checks if URLs, VCS links and email address domains found therein are valid.

9.2.7 adequate

`adequate` checks packages installed on the system and reports policy violations.

The following checks are currently implemented:

- broken symlinks
- missing copyright file
- obsolete conffiles
- Python modules not byte-compiled
- missing libraries, undefined symbols, symbol size mismatches
- program name collisions
- missing alternatives
- missing `binfmt` interpreters and detectors
- missing `pkg-config` dependencies
- invalid user/group values in systemd/D-Bus/sysvinit files.

See /usr/share/doc/adequate/README on how to have `adequate` be ran automatically using `autopkgtest`.

9.2.8 i18nspector

i18nspector is a tool for checking translation templates (POT), message catalogues (PO) and compiled message catalogues (MO) files for common problems.

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 attempts to determine the license that applies to each file passed to it, by searching the start of the file for text belonging to various licenses.

9.2.11 blhc

blhc is a tool which checks build logs for missing hardening flags.

9.3 Помощники для debian/rules

Инструменты сборки пакетов делают написание файла debian/rules значительно проще. Дополнительную информацию о том, почему желательно или не желательно использовать их, см. в *Сценарии-помощники*.

9.3.1 debhelper

debhelper is a collection of programs that can be used in debian/rules to automate common tasks related to building binary Debian packages. debhelper includes programs to install various files into your package, compress files, fix file permissions, and integrate your package with the Debian menu system.

Unlike some approaches, debhelper is broken into several small, simple commands, which act in a consistent manner. As such, it allows more fine-grained control than some of the other debian/rules tools.

Имеется ряд небольших дополнительных пакетов debhelper, которые слишком мелки, чтобы описывать их здесь. Вы можете посмотреть список этих программ, выполнив apt-cache search ^dh-.

When choosing a debhelper compatibility level for your package, you should choose the highest compatibility level that is supported in the most recent stable release. Only use a higher compatibility level if you need specific features that are provided by that compatibility level that are not available in earlier levels.

In the past the compatibility level was defined in debian/compat, however nowadays it is much better to not use that but rather to use a versioned build-dependency like debhelper-compat (=12).

9.3.2 dh-make

The dh-make package contains dh_make, a program that creates a skeleton of files necessary to build a Debian package out of a source tree. As the name suggests, dh_make is a rewrite of debmake, and its template files use dh_* programs from debhelper.

Хотя файлы rules, порождаемые dh_make, вообще-то являются достаточной основой для создания рабочего пакета, они всё равно представляют собой лишь основу: на сопровождающем всё ещё лежит груз задачи по тонкой настройке порождённых файлов и сборке пакета, целиком соответствующего Политике и работающего.

9.3.3 equivs

`equivs` представляет собой ещё один пакет для создания пакетов. Часто он предлагается для локального использования, если вам нужно сделать пакет просто для удовлетворения зависимостей. Также он иногда используется при создании „метапакетов“, это пакеты, чья цель состоит лишь в том, чтобы зависеть от других пакетов.

9.4 Сборщики пакетов

The following packages help with the package building process, general driving of `dpkg-buildpackage`, as well as handling supporting tasks.

9.4.1 git-buildpackage

`git-buildpackage` предоставляет возможность введения или импорта пакетов Debian с исходным кодом в репозиторий Git, сборки пакета Debian из репозитория Git, а также помогает в интеграции изменений из основной ветки разработки в этот репозиторий.

Эти утилиты предоставляют инфраструктуру для облегчения использования Git сопровождающими Debian. Это позволяет хранить отдельные ветки Git пакета для `стабильного`, `нестабильного` и возможно `экспериментального` выпусков, а также иметь все другие преимущества системы контроля версий.

9.4.2 debootstrap

Пакет `debootstrap` и соответствующий сценарий позволяют вам произвести начальную установку базовой системы Debian в любую часть файловой системы. Под базовой системой мы подразумеваем минимальное число пакетов, необходимых для работы и установки остальной системы.

Иметь подобную систему весьма полезно. Например, вы можете сделать `chroot` в эту систему и проверить ваши сборочные зависимости. Либо вы можете проверить то, как ведут себя ваши пакеты при установке на базовую систему. Сборщики chroot используют этот пакет; см. об этом ниже.

9.4.3 pbuilder

`pbuilder` constructs a chrooted system, and builds a package inside the chroot. It is very useful to check that a package's build dependencies are correct, and to be sure that unnecessary and wrong build dependencies will not exist in the resulting package.

A related package is `cowbuilder`, which speeds up the build process using a COW filesystem on any standard Linux filesystem.

9.4.4 sbuild

`sbuild` представляет собой другой автоматизированный сборщик. Он также может использовать окружения chroot. Он может использоваться как отдельно, так и как часть сетевого, распределённого окружения. В последнем случае он является частью системы, используемой теми, кто занимается переносом, для сборки двоичных пакетов для всех доступных архитектур. Дополнительную информацию см. в *wanna-build*, систему в действии можно посмотреть по адресу https://buildd.debian.org/.

9.5 ПО для загрузки пакетов

Следующие пакеты помогут автоматизировать или упростить процесс загрузки пакетов в официальный архив.

9.5.1 dupload

`dupload` is a package and a script to automatically upload Debian packages to the Debian archive, to log the upload, and to optionally send mail about the upload of a package. It supports various kinds of hooks to extend its functionality, and can be configured for new upload locations or methods, although by default it provides various hooks performing checks and comes configured with all Debian upload locations.

9.5.2 dput

The `dput` package and script do much the same thing as `dupload`, but in a different way. Out of the box it supports to run `dinstall` in dry-run mode after the upload.

9.5.3 dcut

Сценарий `dcut` (часть пакета `dput`, *dput*) помогает удалять файлы из каталога загрузки на ftp.

9.6 Автоматизация сопровождения пакетов

Следующие инструменты помогают автоматизировать различные задачи по сопровождению пакетов от добавления записей в журнал изменений или строк подписи и поиска ошибок в Emacs до использования наиболее свежего и исключительно официального файла `config.sub`.

9.6.1 devscripts

`devscripts` is a package containing wrappers and tools that are very helpful for maintaining your Debian packages. Example scripts include `debchange` (or its alias, `dch`), which manipulates your `debian/changelog` file from the command-line, and `debuild`, which is a wrapper around `dpkg-buildpackage`. The `bts` utility is also very helpful to update the state of bug reports on the command line. `uscan` can be used to watch for new upstream versions of your packages (see https://wiki.debian.org/debian/watch for more info on that). `suspicious-source` outputs a list of files which are not common source files.

See the devscripts(7) manual page for a complete list of available scripts.

9.6.2 reportbug

`reportbug` is a tool designed to make the reporting of bugs in Debian and derived distributions relatively painless. Its features include:

- Integration with mutt and mh/nmh mail readers.
- Access to outstanding bug reports to make it easier to identify whether problems have already been reported.
- Automatic checking for newer versions of packages.

`reportbug` is designed to be used on systems with an installed mail transport agent; however, you can edit the configuration file and send reports using any available mail server.

This package also includes the `querybts` script for browsing the Debian bug tracking system.

9.6.3 autotools-dev

`autotools-dev` contains best practices for people who maintain packages that use `autoconf` and/or `automake`. Also contains canonical `config.sub` and `config.guess` files, which are known to work on all Debian ports.

9.6.4 dpkg-repack

`dpkg-repack` creates a Debian package file out of a package that has already been installed. If any changes have been made to the package while it was unpacked (e.g., files in `/etc` were modified), the new package will inherit the changes.

This utility can make it easy to copy packages from one computer to another, or to recreate packages that are installed on your system but no longer available elsewhere, or to save the current state of a package before you upgrade it.

9.6.5 alien

`alien` преобразует пакеты между различными форматами пакетов, включая пакеты Debian, RPM (RedHat), LSB (Linux Standard Base), Solaris, и 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` (из пакета `devscripts`, *devscripts*) запускает команду в окружении `strace` для определения всех пакетов, которые используются вызванной командой.

Для пакетов Debian это весьма полезно, если вам необходимо сформировать строку `Build-Depends` для вашего нового пакета: запуск процесса сборки через `dpkg-depcheck` предоставит вам приблизительный список сборочных зависимостей. Например:

```
dpkg-depcheck -b debian/rules build
```

`dpkg-depcheck` также может использоваться для проверки зависимостей времени исполнения, особенно в том случае, если ваш пакет использует exec 2 для запуска других программ.

Дополнительную информацию см. в dpkg-depcheck 1.

9.6.8 debputy

The `debputy` tools is new since 2024. While its main purpose is to offer a new Debian package build paradigm, it includes subcommands that can be used on any existing Debian package to validate the correctness of most of the files in `debian/*`, and in many cases also automatically fix them.

To check correctness of files in debian/* run:

```
debputy lint --spellcheck
```

To format debian/control and debian/tests/control files

```
debputy reformat --style black
```

Using the `reformat` command obsoletes using `wrap-and-sort -ast`.

The debputy tool also includes a language server which, when integrated with a code editor, can give real-time feedback on the correctness of files in `debian/*` while editing them.

For more information please see debputy 1.

9.7 Инструменты для переноса

Следующие инструменты полезны для занимающихся переносом и кросс-компиляцией.

9.7.1 dpkg-cross

`dpkg-cross` является инструментом для установки библиотек и заголовочных файлов для перекрёстной компиляции схожим с `dpkg` способом. Более того, функциональность `dpkg-buildpackage` и `dpkg-shlibdeps` была улучшена в плане поддержки перекрёстной компиляции.

9.8 Документация и информацию

Следующие пакеты предоставляют информацию для сопровождающих, либо помогают им в сборке документации.

9.8.1 debian-policy

The `debian-policy` package contains the Debian Policy Manual and related documents, which are:

- Debian Policy Manual
- Filesystem Hierarchy Standard (FHS)
- Debian Menu sub-policy
- Debian Perl sub-policy
- Debian configuration management specification
- Machine-readable debian/copyright specification
- Autopkgtest - automatic as-installed package testing
- Authoritative list of virtual package names
- Policy checklist for upgrading your packages

The Debian Policy Manual the policy relating to packages and details of the packaging mechanism. It covers everything from required `gcc` options to the way the maintainer scripts (`postinst` etc.) work, package sections and priorities, etc.

Also useful is the file `/usr/share/doc/debian-policy/upgrading-checklist.txt.gz`, which lists changes between versions of policy.

9.8.2 doc-debian

`doc-debian` contains lots of useful Debian-specific documentation:

- Debian Linux Manifesto
- Constitution for the Debian Project
- Debian Social Contract
- Debian Free Software Guidelines
- Debian Bug Tracking System documentation
- Introduction to the Debian mailing lists

9.8.3 developers-reference

The `developers-reference` package contains the document you are reading right now, the Debian Developer's Reference, a set of guidelines and best practices which has been established by and for the community of Debian developers.

9.8.4 maint-guide

The `maint-guide` package contains the Debian New Maintainers' Guide.

This document tries to describe the building of a Debian package to ordinary Debian users and prospective developers. It uses fairly non-technical language, and it's well covered with working examples.

9.8.5 debmake-doc

The `debmake-doc` package contains the Guide for Debian Maintainers.

This document is newer than Debian New Maintainers' Guide and intends to replace it. The Guide for Debian Maintainers caters to those learning Debian packaging and covers a wide range of topics and tools, along with plenty of examples about various types of packaging issues.

9.8.6 packaging-tutorial

This tutorial is an introduction to Debian packaging. It teaches prospective developers how to modify existing packages, how to create their own packages, and how to interact with the Debian community.

In addition to the main tutorial, it includes three practical sessions on modifying the `grep` package, and packaging the `gnujump` game and a Java library.

9.8.7 how-can-i-help

`how-can-i-help` shows opportunities for contributing to Debian. `how-can-i-help` hooks into `APT` to list opportunities for contributions to Debian (orphaned packages, bugs tagged 'newcomer') for packages installed locally, after each `APT` invocation. It can also be invoked directly, and then lists all opportunities for contribution (not just the new ones).

9.8.8 docbook-xml

`docbook-xml` provides the DocBook XML DTDs, which are commonly used for Debian documentation (as is the older debiandoc SGML DTD).

Пакет `docbook-xsl` предоставляет файлы XSL для сборки и форматирования стиля исходных файлов в различные форматы вывода. Для того, чтобы использовать стили XSL, вам потребуется процессор XSLT, такой как `xsltproc`. Документация для стилей может быть найдена в пакетах `docbook-xsl-doc-*`.

To produce PDF from FO, you need an FO processor, such `fop` or `dblatex`.

9.8.9 debiandoc-sgml

`debiandoc-sgml` provides the DebianDoc SGML DTD, which has been commonly used for Debian documentation, but is now deprecated (`docbook-xml` or `python3-sphinx` should be used instead).

9.8.10 debian-keyring

Contains the public OpenPGP keys of Debian Developers and Maintainers. See *Сопровождение вашего открытого ключа* and the package documentation for more information.

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.