Debian Developer's Reference

Release 14.18

Developer's Reference Team

2026-11-02

Indice

1 Scopo di questo documento **3**

2 Applying to Become a Member **5**
2.1 Iniziare . . . 5
2.2 Mentor e sponsor Debian . . . 5
2.3 Registering as a Debian member . . . 6

3 Doveri di un Debian Developer **9**
3.1 Doveri di un maintainer di pacchetti . . . 9
3.1.1 Lavorare per il prossimo rilascio `stable` . . . 9
3.1.2 Mantenere i pacchetti in `stable` . . . 9
3.1.3 Gestire i bug critici per il rilascio . . . 9
3.1.4 Coordinamento con gli sviluppatori originali . . . 10
3.2 Doveri amministrativi . . . 10
3.2.1 Gestire le vostre informazioni Debian . . . 10
3.2.2 Mantenere la vostra chiave pubblica . . . 11
3.2.3 Votare . . . 11
3.2.4 Decision making . . . 11
3.2.5 Andare in vacanza con garbo . . . 11
3.2.6 Congedarsi . . . 12
3.2.7 Ritornare dopo il congedo . . . 12

4 Resources for Debian Members **13**
4.1 Mailing list . . . 13
4.1.1 Regole di base per l'utilizzo . . . 13
4.1.2 Mailing list dello sviluppo centrale . . . 13
4.1.3 Mailing list speciali . . . 14
4.1.4 Richiedere nuove liste connesse con lo sviluppo . . . 14
4.2 Canali IRC . . . 14
4.3 La documentazione . . . 14
4.4 Le macchine Debian . . . 15
4.4.1 Il server dei bug . . . 15
4.4.2 Il server ftp-master . . . 15
4.4.3 Il server www-master . . . 15
4.4.4 Il web server persone . . . 16
4.4.5 salsa.debian.org: Git repositories and collaborative development platform . . . 16
4.4.6 GitHub.com: Submitting pull requests to upstream repositories . . . 16
4.5 Il Database degli sviluppatori . . . 16
4.6 L'archivio Debian . . . 17
4.6.1 Sezioni . . . 18
4.6.2 Le architetture . . . 19

4.6.3 I pacchetti . . . 19
4.6.4 Le distribuzioni . . . 19
4.6.4.1 Stable, testing e unstable . . . 20
4.6.4.2 Maggiori informazioni sulla distribuzione testing . . . 20
4.6.4.3 Experimental . . . 20
4.6.5 Nomi in codice dei rilasci . . . 21
4.7 Mirror di Debian . . . 21
4.8 Il sistema Incoming . . . 22
4.9 Informazioni sul pacchetto . . . 22
4.9.1 Sul web . . . 22
4.9.2 L'utility `dak ls` . . . 22
4.10 The Debian Package Tracker . . . 23
4.11 Panoramica dei pacchetti per sviluppatori . . . 23
4.12 L'installazione FusionForge di Debian: Alioth . . . 24
4.13 Goodies for Debian Members . . . 24

5 Gestione dei pacchetti **25**
5.1 Nuovi pacchetti . . . 25
5.2 Registrare i cambiamenti nel pacchetto . . . 26
5.3 Testare del pacchetto . . . 26
5.4 Struttura del pacchetto sorgente . . . 27
5.5 Scegliere una distribuzione . . . 27
5.5.1 Caso particolare: caricamenti sulle distribuzioni `stable` e `oldstable` . . . 27
5.5.2 Special case: the `stable-updates` suite . . . 28
5.5.3 Caso particolare: caricamenti su `testing/testing-proposed-updates` . . . 29
5.6 Caricare un pacchetto . . . 29
5.6.1 Source and binary uploads . . . 29
5.6.2 Caricamento su `ftp-master` . . . 30
5.6.3 Caricamenti differiti . . . 30
5.6.4 Caricamenti di sicurezza . . . 30
5.6.5 Altre code di caricamento . . . 30
5.6.6 Notifications . . . 30
5.7 Specificare la sezione del pacchetto, sottosezione e la priorità . . . 31
5.8 New upstream versions . . . 31
5.9 Gestione dei bug . . . 32
5.9.1 Monitoraggio dei bug . . . 32
5.9.2 Rispondere ai bug . . . 32
5.9.3 Pulizia dei bug . . . 33
5.9.4 Quando i bug vengono chiusi da nuovi upload . . . 34
5.9.5 Gestione di bug relativi alla sicurezza . . . 35
5.9.5.1 Debian Security Tracker . . . 35
5.9.5.2 Riservatezza . . . 35
5.9.5.3 Avvisi di sicurezza . . . 36
5.9.5.4 Preparazione di pacchetti per indirizzare i problemi di sicurezza . . . 36
5.9.5.5 Caricamento del pacchetto corretto . . . 37
5.10 Subscribing to package updates . . . 38
5.11 Lo spostamento, la rimozione, la ridenominazione, l'adozione, e rendere orfani i pacchetti . . . 38
5.11.1 Spostare i pacchetti . . . 38
5.11.2 Rimozione dei pacchetti . . . 38
5.11.2.1 Rimozione di pacchetti da `Incoming` . . . 39
5.11.3 La sostituzione o la ridenominazione dei pacchetti . . . 39
5.11.4 Pacchetto orfano . . . 40
5.11.5 L'adozione di un pacchetto . . . 40
5.11.6 Reintrodurre pacchetti . . . 40
5.12 Fare port e port del proprio lavoro . . . 41
5.12.1 Siate gentili con gli autori di port . . . 41
5.12.2 Linee guida per i caricamenti degli autori di port . . . 42
5.12.2.1 Ricompilazione o NMU dei soli binari . . . 42

5.12.2.2 Quando fare un NMU del sorgente se si è un autore di port . . . 43
5.12.3 Infrastrutture e automazione per il port . . . 43
5.12.3.1 Mailing list e pagine web . . . 43
5.12.3.2 Strumenti per gli autori di port . . . 44
5.12.3.3 `wanna-build` . . . 44
5.12.4 Quando il pacchetto *non* è portabile . . . 44
5.12.5 Marcare i pacchetti non-free come compilabili automaticamente . . . 45
5.13 Caricamenti dei Non-Maintainer (NMU) . . . 45
5.13.1 Quando e come fare un NMU . . . 45
5.13.2 NMU e `debian/changelog` . . . 46
5.13.3 Utilizzare la coda `DELAYED/` . . . 47
5.13.4 NMU dal punto di vista del maintainer . . . 47
5.13.5 Source NMUs vs Binary-only NMUs (binNMUs) . . . 47
5.13.6 NMU e caricamenti di QA . . . 48
5.13.7 NMU e caricamenti del team . . . 48
5.14 Package Salvaging . . . 48
5.14.1 When a package is eligible for package salvaging . . . 49
5.14.2 How to salvage a package . . . 49
5.15 La manutenzione collaborativa . . . 50
5.16 La distribuzione testing . . . 50
5.16.1 Nozioni di base . . . 50
5.16.2 Aggiornamenti da unstable . . . 50
5.16.2.1 Obsoleti . . . 51
5.16.2.2 Rimozioni da testing . . . 52
5.16.2.3 Dipendenze circolari . . . 52
5.16.2.4 Influenza del pacchetto in testing . . . 52
5.16.2.5 Dettagli . . . 52
5.16.3 Aggiornamenti diretti di testing . . . 53
5.16.4 Domande frequenti . . . 53
5.16.4.1 Quali sono i bug critici per il rilascio e come vengono contati? . . . 53
5.16.4.2 Come potrebbe l'installazione di un pacchetto in `testing` rendere difettosi altri pacchetti? . . . 53
5.17 The Stable backports archive . . . 54
5.17.1 Nozioni di base . . . 54
5.17.2 Exception to the testing-first rule . . . 54
5.17.3 Who can maintain packages in the stable-backports archive? . . . 54
5.17.4 When can one start uploading to stable-backports? . . . 54
5.17.5 How long must a package be maintained when uploaded to stable-backports? . . . 54
5.17.6 How often shall one upload to stable-backports? . . . 55
5.17.7 How can one learn more about backporting? . . . 55

6 Buone pratiche per la pacchettizzazione **57**
6.1 Buone pratiche per `debian/rules` . . . 57
6.1.1 Script di supporto . . . 57
6.1.2 Pacchetti binari multipli . . . 58
6.2 Buone pratiche per `debian/control` . . . 58
6.2.1 The package name . . . 58
6.2.2 Linee guida generali per le descrizioni dei pacchetti . . . 58
6.2.3 La sinossi del pacchetto, o una breve descrizione . . . 59
6.2.4 La descrizione lunga . . . 59
6.2.5 Home page originale del pacchetto . . . 60
6.2.6 La posizione del Version Control System . . . 60
6.2.6.1 Vcs-Browser . . . 60
6.2.6.2 Vcs-* . . . 60
6.3 Buone pratiche per il `debian/changelog` . . . 61
6.3.1 Scrivere informazioni utili nel file changelog . . . 61
6.3.2 Selecting the upload urgency . . . 61
6.3.3 Comuni incomprensioni sulle voci del changelog . . . 62

6.3.4 Errori frequenti nelle voci del changelog . . . 62
6.3.5 Integrare i changelog con i file NEWS.Debian . . . 63
6.4 Best practices around security . . . 63
6.5 Buone pratiche per gli script del mantainer . . . 63
6.5.1 Generic guidelines . . . 63
6.5.2 Expected behavior of complex postinst scripts . . . 64
6.6 Gestione della configurazione con debconf . . . 64
6.6.1 Non abusare di debconf . . . 64
6.6.2 Raccomandazioni generali per autori e traduttori . . . 65
6.6.2.1 Scrivere inglese corretto . . . 65
6.6.2.2 Sii gentile con i traduttori . . . 65
6.6.2.3 Ordinare intere traduzioni quando si correggono errori di battitura e di ortografia 65
6.6.2.4 Non fare ipotesi sulle interfacce . . . 66
6.6.2.5 Non utilizzare la prima persona . . . 66
6.6.2.6 Usare il genere neutro . . . 66
6.6.3 Definizione dei campi dei modelli . . . 66
6.6.3.1 Tipo . . . 67
6.6.3.2 Descrizione: descrizione breve ed estesa . . . 68
6.6.3.3 Scelte . . . 68
6.6.3.4 Default . . . 68
6.6.4 Template fields specific style guide . . . 68
6.6.4.1 Type field . . . 68
6.6.4.2 Il campo Description . . . 68
6.6.4.3 Campo Choises . . . 69
6.6.4.4 Campo Default . . . 69
6.7 Internazionalizzazione . . . 70
6.7.1 Gestione delle traduzioni debconf . . . 70
6.7.2 Documentazione internazionalizzata . . . 70
6.8 Best practices for debian/patches . . . 71
6.9 Situazioni comuni durante la pacchettizzazione . . . 71
6.9.1 Pacchettizzazione utilizzando autoconf/automake . . . 71
6.9.2 Le librerie . . . 71
6.9.3 La documentazione . . . 71
6.9.4 Specifici tipi di pacchetti . . . 72
6.9.5 Dati indipendenti dall'architettura . . . 72
6.9.6 Aver bisogno di una particolare localizzazione durante la compilazione . . . 73
6.9.7 Rendere i pacchetti di transizione conformi a deborphan . . . 73
6.9.8 Buone pratiche per i file .orig.tar.{gz,bz2,xz} . . . 73
6.9.8.1 Sorgente puro . . . 73
6.9.8.2 Sorgente originale ripacchettizzato . . . 74
6.9.8.3 Modifica dei file binari . . . 75
6.9.9 Buone pratiche per i pacchetti di debug . . . 75
6.9.9.1 Automatically generated debug packages . . . 75
6.9.9.2 Manual -dbg packages . . . 75
6.9.10 Buone pratiche per i metapacchetti . . . 76

7 Oltre la pacchettizzazione **77**
7.1 Segnalare i bug . . . 77
7.1.1 Riportare molti bug in una sola volta (presentazione massiccia di bug) . . . 78
7.1.1.1 Usertag . . . 78
7.2 Costo della Quality Assurance . . . 78
7.2.1 Lavoro giornaliero . . . 78
7.2.2 Bug squashing parties . . . 79
7.3 Come contattare gli altri maintainer . . . 79
7.4 Rapportarsi con maintainer non attivi e/o non raggiungibili . . . 79
7.5 Dealing with repeatedly negligent Debian developers . . . 80
7.6 Interagire con potenziali sviluppatori Debian . . . 80
7.6.1 Sponsorizzare pacchetti . . . 81

7.6.1.1 Sponsorizzare un nuovo pacchetto . . . 81
7.6.1.2 Sponsorizzare un aggiornamento di un pacchetto esistente . . . 82
7.6.2 Granting upload permissions to DMs . . . 83
7.6.3 Promuovere nuovi sviluppatori . . . 83
7.6.4 Gestire nuove candidature di maintainer . . . 83

8 Internazionalizzazione e traduzioni **85**
8.1 Come le traduzioni sono effettuate in Debian . . . 85
8.2 I18N e L10N FAQ per i maintainer . . . 86
8.2.1 Come ottenere un certo testo tradotto . . . 86
8.2.2 Come ottenere una revisione di una data traduzione . . . 86
8.2.3 Come ottenere una data traduzione aggiornata . . . 86
8.2.4 Come gestire una segnalazione di bug riguardante una traduzione . . . 86
8.3 I18N & L10N FAQ per traduttori . . . 86
8.3.1 Come aiutare lo sforzo di traduzione . . . 86
8.3.2 Come fornire una traduzione per l' inclusione in un pacchetto . . . 87
8.4 L'attuale pratica consigliata riguardanti l'l10n . . . 87

9 Panoramica degli strumenti del Debian Maintainer **89**
9.1 Strumeti di base . . . 89
9.1.1 `dpkg-dev` . . . 89
9.1.2 `debconf` . . . 89
9.1.3 `fakeroot` . . . 90
9.2 Strumenti per la pulizia di pacchetti . . . 90
9.2.1 `lintian` . . . 90
9.2.2 `lintian-brush` . . . 90
9.2.3 `piuparts` . . . 90
9.2.4 `debdiff` . . . 90
9.2.5 `diffoscope` . . . 91
9.2.6 `duck` . . . 91
9.2.7 `adequate` . . . 91
9.2.8 `i18nspector` . . . 91
9.2.9 `cme` . . . 91
9.2.10 `licensecheck` . . . 91
9.2.11 `blhc` . . . 91
9.3 Strumenti ausiliari per `debian/rules` . . . 92
9.3.1 `debhelper` . . . 92
9.3.2 `dh-make` . . . 92
9.3.3 `equivs` . . . 92
9.4 Strumenti di compilazione . . . 92
9.4.1 `git-buildpackage` . . . 92
9.4.2 `debootstrap` . . . 93
9.4.3 `pbuilder` . . . 93
9.4.4 `sbuild` . . . 93
9.5 Strumenti per caricare i pacchetti . . . 93
9.5.1 `dupload` . . . 93
9.5.2 `dput` . . . 93
9.5.3 `dcut` . . . 93
9.6 Automazione della manutenzione . . . 93
9.6.1 `devscripts` . . . 93
9.6.2 `reportbug` . . . 94
9.6.3 `autotools-dev` . . . 94
9.6.4 `dpkg-repack` . . . 94
9.6.5 `alien` . . . 94
9.6.6 `elpa-dpkg-dev-el` . . . 94
9.6.7 `dpkg-depcheck` . . . 94
9.6.8 `debputy` . . . 95
9.7 Strumenti per i port . . . 95

9.7.1 dpkg-cross . . . 95
9.8 Documentazione ed informazioni . . . 95
9.8.1 debian-policy . . . 95
9.8.2 doc-debian . . . 96
9.8.3 developers-reference . . . 96
9.8.4 maint-guide . . . 96
9.8.5 debmake-doc . . . 96
9.8.6 packaging-tutorial . . . 96
9.8.7 how-can-i-help . . . 96
9.8.8 docbook-xml . . . 96
9.8.9 debiandoc-sgml . . . 97
9.8.10 debian-keyring . . . 97
9.8.11 elpa-debian-el . . . 97

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.

CAPITOLO 1

Scopo di questo documento

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 (*Nuovi pacchetti*) and how to upload packages (*Caricare un pacchetto*); how to handle bug reports (*Gestione dei bug*); how to move, remove, or orphan packages (*Lo spostamento, la rimozione, la ridenominazione, l'adozione, e rendere orfani i pacchetti*); how to port packages (*Fare port e port del proprio lavoro*); and how and when to do interim releases of other maintainers' packages (*Caricamenti dei Non-Maintainer (NMU)*).

Le risorse discusse in questo manuale includono le mailing list (*Mailing list*) e server (*Le macchine Debian*); una discussione sulla struttura dell'archivio Debian (*L'archivio Debian*); spiegazione dei diversi server che accettano caricamenti di pacchetti (*Caricamento su ftp-master*); e una discussione delle risorse che possono aiutare i maintainer con la qualità dei propri pacchetti (*Panoramica degli strumenti del Debian Maintainer*).

Dovrebbe essere chiaro che questo manuale non tratta i dettagli tecnici di pacchetti Debian né come generarli. Né questo manuale dettaglia le norme che il software Debian deve rispettare. Tutte queste informazioni possono essere trovate nel Debian Policy Manual.

Inoltre, questo documento *non è espressione della policy ufficiale*. Contiene documentazione per il sistema Debian e le migliori pratiche più condivise. Pertanto, non è quello che viene chiamato un documento «normativo».

CAPITOLO 2

Applying to Become a Member

2.1 Iniziare

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 *Mailing list*. `debian-devel-announce@lists.debian.org` is another list, which is mandatory for anyone who wishes to follow Debian's development.

È possibile iscriversi ed osservare (cioè leggere senza inviare) per un po' prima di fare qualsiasi codifica, e si dovrebbero pubblicare le proprie intenzioni di lavorare su qualcosa per evitare la duplicazione degli sforzi.

Un'altra buona lista da sottoscrivere è `debian-mentors@lists.debian.org`. Si consulti *Mentor e sponsor Debian* per i dettagli. Il canale IRC `#debian` può anche essere utile, si consulti *Canali IRC*.

Quando si sa come si vuole contribuire a Debian, si dovrebbe entrare in contatto con i maintainer Debian esistenti che lavorano su compiti simili. In questo modo, si può imparare da sviluppatori esperti. Per esempio, se si è interessati nella pacchettizzazione del software esistente per Debian, si dovrebbe cercare di ottenere uno sponsor. Uno sponsor lavorerà insieme sul proprio pacchetto e caricherà nell'archivio Debian una volta che sarà contento del lavoro di pacchettizzazione che si è fatto. È possibile trovare uno sponsor inviando una email alla mailing list `debian-mentors@lists.debian.org`, descrivendo il pacchetto e voi stessi e chiedendo uno sponsor (si veda *Sponsorizzare pacchetti* e https://wiki.debian.org/DebianMentorsFaqper ulteriori informazioni sulla sponsorizzazione). D'altra parte, se si è interessati al porting di Debian per architetture o kernel alternativi è possibile iscriversi a mailing list specifiche e chiedere lì come iniziare. Infine, se si è interessati alla documentazione o alla Quality Assurance (QA) ci si può unire ai maintainer che già lavorano su questi compiti e inviare patch e miglioramenti.

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 Mentor e sponsor Debian

La mailing list `debian-mentors@lists.debian.org` è stata istituita per i maintainer alle prime armi che cercano aiuto con l'iniziale pacchettizzazione e altri problemi legati allo sviluppo. Ogni nuovo sviluppatore è invitato a iscriversi a questa lista (si consulti *Mailing list* per i dettagli).

Coloro che preferiscono un aiuto direttamente da un'altra persona (ad esempio, attraverso email privata) dovrebbero anche pubblicare in quella lista e uno sviluppatore esperto volontariamente aiuterà.

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.

Se volete essere un mentore e/o uno sponsor, maggiori informazioni sono disponibili in *Interagire con potenziali sviluppatori 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.

Prima che realmente ci si registri si dovrebbe aver dimostrato che si può fare un lavoro competente e che si sarà un buon collaboratore. Si mostrerà ciòmandando patch attraverso il Bug Tracking System e avendo per un po' un pacchetto sponsorizzato da uno sviluppatore Debian esistente. Inoltre, ci si aspetta che i collaboratori siano interessati a tutto il progetto e non solo a mantenere i propri pacchetti. Se si può aiutare gli altri maintainer, fornendo ulteriori informazioni su un bug o anche una patch, lo si faccia!

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 *Mantenere la vostra chiave pubblica* 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.

Se la chiave pubblica non è su un server a chiave pubblica, come `subkeys.pgp.net`, leggere la documentazione disponibile presso NM Fase 2: Identificazione. Questo documento contiene le istruzioni su come mettere la vostra chiave sui server di chiavi pubbliche. Il New Maintainer Group metterà la vostra chiave pubblica sul server, se non è già presente.

Alcuni paesi limitano l'utilizzo di software di crittografia per i loro cittadini. Questo non deve impedire tuttavia le attività come maintainer Debian di un pacchetto, essendo perfettamente legale l'utilizzo di prodotti di crittografia per l'autenticazione, piuttosto che per scopi di crittografia. Se si vive in un paese dove è vietato l'uso della crittografia per l'autenticazione, allora non esitare a contattarci in modo da poter prendere accordi speciali.

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.

CAPITOLO 3

Doveri di un Debian Developer

3.1 Doveri di un maintainer di pacchetti

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 Lavorare per il prossimo rilascio `stable`

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.

Più concretamente, si dovrà controllare che i pacchetti stiano migrando verso `testing` (si consulti *La distribuzione testing*). Quando la migrazione non avviene dopo il periodo di prova, si deve analizzare il motivo e lavorare per correggerlo. Potrebbe significare correggere il pacchetto (nel caso dei bug critici per il rilascio o di fallimenti nella compilazione su alcune architetture), ma potrebbe anche significare l'aggiornamento (o di correzione, o di rimozione da `testing`) di altri pacchetti per aiutare il completamento di una transizione in cui il proprio pacchetto è incastrato a causa delle sue dipendenze. Il team di rilascio potrebbe fornire alcuni input sugli attuali blocchi di una data transizione, se non si è in grado di identificarli.

3.1.2 Mantenere i pacchetti in `stable`

La maggior parte del lavoro del maintainer del pacchetto è fornire versioni aggiornate dei pacchetti in `unstable`, ma il loro lavoro comporta anche prendersi cura dei pacchetti nell'attuale rilascio `stable`.

Mentre i cambiamenti in `stable` sono scoraggiati, sono comunque possibili. Ogni volta che un problema di sicurezza è segnalato, si dovrebbe collaborare con il team di sicurezza per fornire una versione corretta (si veda *Gestione di bug relativi alla sicurezza*). Quando i bug di gravità importante (o più) vengono segnalati per la versione `stable` dei propri pacchetti, si dovrebbe valutare la possibilità di fornire una correzione mirata. Si potrà chiedere al team di rilascio `stable` se accettano tale tipo di aggiornamento e poi preparare un caricamento `stable` (si consulti *Caso particolare: caricamenti sulle distribuzioni stable e oldstable*).

3.1.3 Gestire i bug critici per il rilascio

In generale si dovrebbe affrontare le segnalazioni di bug sui propri pacchetti come è descritto in *Gestione dei bug*. Tuttavia, c'è una categoria speciale di bug di cui vi dovrete prendere cura - i cosiddetti bug critici per il rilascio (RC bug). Tutte le segnalazioni di bug che hanno gravità `critical`, `grave` o `serious` rendono il pacchetto non adatto per l'inclusione nel prossimo rilascio `stable`. Quindi possono ritardare il rilascio di Debian (quando riguardano

un pacchetto in `testing`) o bloccare migrazioni in `testing` (quando influiscono solo sul pacchetto in `unstable`). Nello scenario peggiore, procederanno alla rimozione del pacchetto. Ecco perché questi bug devono essere corretti al più presto.

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 *Caricamenti dei Non-Maintainer (NMU)*) because they can block the `testing` migration of many packages.

La mancanza di attenzione per i bug RC è spesso interpretata dal team QA come un segno che il maintainer è scomparso senza aver correttamente reso orfano il suo pacchetto. Il team MIA potrebbe anche mettersi in gioco, che potrebbe concretizzarsi nel rendere orfani i vostri pacchetti (si consulti *Rapportarsi con maintainer non attivi e/o non raggiungibili*).

3.1.4 Coordinamento con gli sviluppatori originali

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.

Anche se non è il proprio lavoro correggere i bug specifici non-Debian, si può liberamente farlo se ne si ha la possibilità. Quando si effettuano queste correzioni, ci si assicuri di trasmetterle anche ai maintainer originali. Utenti e sviluppatori Debian a volte invieranno patch che correggono bug dei sorgenti originali: si dovrebbe valutare e trasmettere queste patch allo sviluppatore originale.

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.

Se si necessita di modificare i sorgenti originali al fine di costruire un pacchetto conforme alla policy, allora si dovrebbe proporre una soluzione accurata agli sviluppatori originali che può essere li applicata, in modo da non dover modificare i sorgenti della prossima versione originale. Qualunque cambiamento necessiti, cerca sempre di non effettuare il fork dai sorgenti originali.

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.

Se si scopre che gli sviluppatori originali sono o diventano ostili verso Debian o la comunità del software libero, si potrebbe riconsiderare la necessità di includere il software in Debian. A volte il costo sociale per la comunità Debian non vale i benefici che il software può portare.

3.2 Doveri amministrativi

Un progetto delle dimensioni di Debian si basa su alcune infrastrutture amministrative per tenere traccia di tutto. Come membro del progetto, si hanno alcuni doveri che assicurano che il tutto continui a funzionare senza problemi.

3.2.1 Gestire le vostre informazioni Debian

C'è un database LDAP contenente le informazioni relative agli sviluppatori Debian su https://db.debian.org/. Si dovrebbe immettere le informazioni li ed aggiornarle appena cambiano. Più in particolare, fare in modo che l'indirizzo al quale la propria email debian.org viene inoltrata sia sempre aggiornato, così come l'indirizzo in cui si hanno le proprie iscrizioni debian private se si sceglie di sottoscriverle.

Per ulteriori informazioni sul database, si consulti *Il Database degli sviluppatori*.

3.2.2 Mantenere la vostra chiave pubblica

Be very careful with your private keys. Do not place them on any public servers or multiuser machines, such as the Debian servers (see *Le macchine 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.

È necessario garantire non solo che la vostra chiave è sicura contro il furto, ma anche che è protetta in caso di smarrimento. Generare e fare una copia (meglio anche se in forma cartacea) del certificato di revoca; questo è necessario se la chiave viene persa.

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.

Si applichino le stesse routine di estrazione di chiavi discusse nel *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 Votare

Anche se Debian non è davvero una democrazia, usiamo un processo democratico per eleggere i nostri leader e ad approvare risoluzioni generali. Queste procedure sono definite dalla Costituzione Debian.

Oltre all'annuale elezione del leader, le votazioni non sono tenute regolarmente e non sono intraprese con leggerezza. Ogni proposta è prima discussa sulla mailing list `debian-vote@lists.debian.org` e richiede diverse approvazioni prima che il segretario del progetto inizii la procedura di voto.

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.

L'elenco di tutte le proposte (passate e presenti) è disponibile sul Debian Voting Information pagina, insieme a informazioni su come fare, supportare e votare proposte.

3.2.4 Decision making

Besides the issues covered in *Votare*, 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 Andare in vacanza con garbo

È comune per gli sviluppatori avere periodi di assenza, se queste sono le vacanze programmate o semplicemente se sono sepolti in altri lavori. La cosa importante da notare è che gli altri sviluppatori hanno bisogno di sapere che

si è in vacanza in modo che possano fare tutto ciò che è necessario in caso di problemi con i propri pacchetti o altri obblighi nel progetto.

Di solito questo significa che altri sviluppatori sono consentiti NMU (si consulti *Caricamenti dei Non-Maintainer (NMU)*) per il proprio pacchetto se un grosso problema (bug critico per la release, aggiornamento della sicurezza, etc.), si verifica mentre si è in vacanza. A volte non è niente di così critico come quelli, ma è ancora il caso di far sapere agli altri che non si è disponibili.

Al fine di informare gli altri sviluppatori, ci sono due cose che si dovrebbero fare. In primo luogo inviare una email a `@debian-private e liste-host;` con [VAC] anteposto all'argomento del messaggio[1] e indicare il periodo di tempo in cui si sarà in vacanza. Si possono anche dare alcune speciali istruzioni su cosa fare in caso di problemi.

L'altra cosa da fare è quella di segnare se stessi come in vacanza nel *Il Database degli sviluppatori* (questa informazione è accessibile solo agli sviluppatori Debian). Non dimenticare di togliere il flag vacanza quando si torna!

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 Congedarsi

Se si sceglie di lasciare il progetto Debian, è necessario assicurarsi di eseguire le seguenti operazioni:

- Orphan all your packages, as described in *Pacchetto orfano*.
- Remove yourself from uploaders for co- or team-maintained packages.
- Se si ricevono mail da un alias e-mail di @debian.org (e.g: press@debian.org) e si desidera essere rimosso, aprire una segnalazione RT per gli Amministratori dei Sistemi Debian. Si invii una e-mail a `admin@rt.debian.org` con la dicitura "Debian RT" da qualche parte nel soggetto indicando da quali alias si desidera esseere rimossi.
- Please remember to also retire from teams, e.g. remove yourself from team wiki pages or salsa groups.
- 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`

È importante che il processo di cui sopra sia seguito, perché trovare sviluppatori inattivi e rendere orfani i loro pacchetti richiede molto tempo e lavoro.

3.2.7 Ritornare dopo il congedo

A retired developer's account is marked as "emeritus" when the process in *Congedarsi* is followed, and "removed" otherwise. Retired developers with an "emeritus" account can get their account re-activated as follows:

- Log on onto nm.debian.org using your salsa credentials.
- Follow the Return from Emeritus Wizard.
- Si passi attraverso un processo di NM accorciato (per garantire che il committente tornando sappia ancora parti importanti della P&P and 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.

[1] Questo è come il messaggio può essere facilmente filtrato da persone che non vogliono leggere avvisi di vacanza.

CAPITOLO 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 Mailing list

Gran parte delle conversazioni tra gli sviluppatori Debian (e gli utenti) sono gestite attraverso una vasta gamma di mailing list che ospitiamo su `lists.debian.org`. Per saperne di più su come iscriversi o cancellarsi, come inviare e come non inviare, dove trovare vecchi post e come cercarli, come contattare i maintainer delle liste e visionare varie altre informazioni sulle mailing list, leggere https://www.debian.org/MailingLists/. Questa sezione tratterà solo gli aspetti delle mailing list che sono di particolare interesse per gli sviluppatori.

4.1.1 Regole di base per l'utilizzo

Quando si risponde ai messaggi della mailing list, non inviare una copia per conoscenza (CC) al mittente originale a meno che non lo richiedano esplicitamente. Chiunque invii messaggi ad una mailing list dovrebbe leggerla per vedere le risposte.

L'invio incrociato (invio dello stesso messaggio a più mailing list) è sconsigliato. Come sempre in rete, tagliare la citazione di articoli a cui si sta rispondendo. In generale, rispettare le consuete convenzioni per l'invio di messaggi.

Si prega di leggere il codice di condotta per ulteriori informazioni. Vale anche la pena di leggere le Debian Community Guidelines.

4.1.2 Mailing list dello sviluppo centrale

Le principali mailing list Debian che gli sviluppatori dovrebbero utilizzare sono:

- `debian-devel-announce@lists.debian.org`, usata per annunciare cose importanti per gli sviluppatori. Tutti gli sviluppatori dovrebbero essere iscritti a questa lista.
- `debian-devel@lists.debian.org`, utilizzata per discutere di vari problemi tecnici legati allo sviluppo.
- `debian-policy@lists.debian.org`, dove vengono discusse e votate le Debian Policy.
- `debian-project@lists.debian.org`, utilizzata per discutere di varie questioni non tecniche legate al progetto.

Ci sono altre mailing list disponibili per una varietà di particolari argomenti, si consulti https://lists.debian.org/ per un elenco.

4.1.3 Mailing list speciali

debian-private@lists.debian.org è una mailing list speciale per discussioni private tra gli sviluppatori Debian. È pensata per essere utilizzata per i messaggi che per qualsiasi ragione non dovrebbero essere pubblicati pubblicamente. Come tale, è una lista a basso traffico, e gli utenti sono invitati a non usare debian-private@lists.debian.org a meno che non sia veramente necessario. Inoltre, *non* inoltrare email da tale lista a nessuno. Archivi di questa lista non sono disponibili sul web per ovvie ragioni, ma è possibile vederli usando il proprio account di shell su master.debian.org e guardando nella directory ~debian/archive/debian-private/.

debian-email@lists.debian.org, è una mailing list speciale usata come raccogli tutto per la corrispondenza relativa a Debian come per contattare gli autori originali su licenze, bug, etc. o discutere il progetto con altri, nei casi in cui potrebbe essere utile avere la discussione archiviata da qualche parte.

4.1.4 Richiedere nuove liste connesse con lo sviluppo

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.

Se si decide che una mailing list regolare su lists.debian.org è davvero ciò che si desidera, si compilii un modulo di richiesta, seguando l'HOWTO.

4.2 Canali IRC

Diversi canali IRC sono dedicati allo sviluppo di Debian. Essi sono principalmente ospitati sulla rete Open and free technology community (OFTC). La voce DNS irc.debian.org è un alias per irc.oftc.net.

Il principale canale di Debian in generale è #debian. Questo è un grande canale generico dove gli utenti possono trovare le notizie recenti nel topic oltre che fornite dai bot. #debian è per chi parla inglese, ci sono anche #debian.de, #debian-fr, #debian-br e altri canali con nomi simili per utenti di altre lingue.

Il canale principale per lo sviluppo di Debian è #debian-devel. È un canale molto attivo; in genere avrà un minimo di 150 persone in ogni momento della giornata. È un canale per le persone che lavorano su Debian, non è un canale di supporto (c'è #debian per quello). È comunque aperto a chiunque voglia dare un'occhiata (e imparare). Il suo topic è generalmente pieno di informazioni interessanti per gli sviluppatori.

Siccome #debian-devel è un canale aperto, non si dovrebbe parlare di problemi che vengono discussi in debian-private@lists.debian.org. C'è un altro canale per questo scopo, si chiama #debian-private ed è protetto da una chiave. Questa chiave è disponibile presso 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...

Esistono anche alcuni canali di sviluppatori non inglesi, per esempio #debian-devel-fr per le persone di lingua francese interessate allo sviluppo di Debian.

Channels dedicated to Debian also exist on other IRC networks.

4.3 La documentazione

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 Le macchine Debian

Debian ha diversi computer che lavorano come server, molti dei quali utilizzati per le funzioni critiche del progetto Debian. La maggior parte delle macchine sono utilizzate per le attività di port e tutte hanno una connessione permanente a Internet.

Alcune delle macchine sono disponibili per essere utilizzate da singoli sviluppatori, a patto che gli seguano le regole stabilite nella Policy Debian per l'utilizzo delle macchine.

In generale, è possibile utilizzare queste macchine come meglio si crede per scopi relativi a Debian. Essere gentili con gli amministratori di sistema, e di non utilizzare tonnellate e tonnellate di spazio su disco, larghezza di banda, o CPU senza prima ottenere l'approvazione degli amministratori di sistema. Di solito queste macchine sono gestite da volontari.

Fare attenzione a proteggere le proprie password e le chiavi SSH Debian installate sulle macchine Debian. Evitare il login o metodi di caricamento che inviano le password su Internet in chiaro, come Telnet, FTP, POP, etc.

Non inserire alcun materiale che non riguarda Debian sui server Debian, se non è stato prima ottenuto il permesso.

L'elenco attuale delle macchine Debian è disponibile presso https://db.debian.org/machines.cgi. Quella pagina web contiene i nomi delle macchine, informazioni di contatto, informazioni su chi può accedere, le chiavi SSH, etc.

Se si ha un problema con il funzionamento di un server Debian, e si pensa che i gestori del sistema devono essere avvisati di ciò, è possibile consultare l'elenco dei problemi aperti nella coda DSA del nostro sistema di tracciamento delle richieste all'indirizzo https://rt.debian.org/ (si può effettuare il login con user «debian», la cui password è disponibile su `master.debian.org:~debian/misc/rt-password`). Per segnalare un nuovo problema, è sufficiente inviare una email all'indirizzo `admin@rt.debian.org` e fare in modo di mettere la stringa «Debian RT» in qualche parte nell'oggetto.

Se si ha un problema con un certo servizio, non legato all'amministrazione del sistema (come ad esempio i pacchetti da rimuovere dall'archivio, suggerimenti per il sito web, etc.), generalmente si segnala un bug su uno «pseudo-pacchetto». Per informazioni su come segnalare bug si consulti *Segnalare i bug*.

Alcuni dei server centrali sono riservati, ma le informazioni sono duplicate su un altro server.

4.4.1 Il server dei bug

`bugs.debian.org` è il percorso canonico per il Bug Tracking System (BTS).

Se si ha intenzione di fare un po' di analisi statistiche o qualche elaborazione dei dati sui bug Debian, questo sarebbe il posto per farlo. Tuttavia si descrivano le intenzioni su `debian-devel@lists.debian.org` prima di attuare qualsiasi cosa, per ridurre inutili duplicazioni di attività o di tempo di elaborazione sprecato.

4.4.2 Il server 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 *Caricare un pacchetto*.

È riservato; un server specchio è disponibile su `mirror.ftp-master.debian.org`.

I problemi con l'archivio FTP Debian generalmente necessitano di essere segnalati come bug nei confronti dello pseudo-pacchetto `ftp.debian.org` o attraverso una email a `ftpmaster@debian.org`, ma si consultino anche le procedure in *Lo spostamento, la rimozione, la ridenominazione, l'adozione, e rendere orfani i pacchetti*.

4.4.3 Il server www-master

Il server web principale è `www-master.debian.org`. Contiene le pagine web ufficiali, il volto di Debian per la maggior parte dei neofiti.

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 Il web server persone

`people.debian.org` è il server utilizzato per creare pagine web personali degli sviluppatori su qualsiasi cosa relativa a Debian.

Se si dispone di alcune informazioni specifiche di Debian che si vuole fornire sul web, è possibile farlo mettendo il materiale nella cartella `public_html` sotto la propria cartella home su `people.debian.org`. Questa sarà accessibile all'URL `http://people.debian.org/~`*proprio-user-id*`/`.

Si consiglia di utilizzare solo questa particolare posizione, perché ne verrà eseguito il backup, mentre su altri host no.

Di solito l'unica ragione per usare un host diverso è quando si ha bisogno di pubblicare materiali soggetti a restrizioni sull'esportazione degli Stati Uniti, nel qual caso è possibile utilizzare uno degli altri server situati al di fuori degli Stati Uniti.

Si invii una email a `debian-devel@lists.debian.org` se si hanno domande.

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 Il Database degli sviluppatori

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.

Gli sviluppatori possono autenticarsi al database per cambiare varie informazioni su se stessi, come ad esempio:

- 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.
- sottoscrizione a debian-private
- se si è in vacanza
- informazioni personali, come il proprio indirizzo, la nazione, la latitudine e la longitudine del luogo in cui si vive per l'uso nella mappa del mondo degli sviluppatori Debian, telefono e fax, IRC nickname e pagina web
- la password e la shell preferita su macchine del progetto Debian

La maggior parte delle informazioni non sono accessibili al pubblico, ovviamente. Per ulteriori informazioni leggere la documentazione online che si può trovare su https://db.debian.org/doc-general.html.

Gli sviluppatori possono inoltre inviare le loro chiavi SSH per essere utilizzate per l'autorizzazione sulle macchine ufficiali di Debian, e persino aggiungere nuove voci DNS *.debian.net. Tali caratteristiche sono documentate su https://db.debian.org/doc-mail.html.

4.6 L'archivio Debian

La distribuzione Debian è composta da molti pacchetti (attualmente circa 40000 pacchetti sorgente) e alcuni file aggiuntivi (come ad esempio la documentazione e le immagini del disco di installazione).

Ecco un esempio di albero di cartelle di un archivio completo di 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/
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/
```

(continues on next page)

(continua dalla pagina precedente)

```
    ...

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

Come si può vedere, la cartella di livello superiore contiene due cartelle, `dists/` e `pool/`. Quest'ultimo è un «punto di raccolta» in cui si trovano effettivamente i pacchetti e che è gestito dal database di manutenzione dell'archivio e dai programmi di accompagnamento. La prima contiene le distribuzioni, `stable`, `testing` e `unstable`. I file `Packages` e `Sources` delle sottocartelle di distribuzione possono fare riferimento ai file in `pool/`. L'albero delle cartelle sotto ciascuna delle distribuzioni è disposto in modo identico. Ciò che viene descritto qui di seguito per `stable` è ugualmente applicabile alle distribuzioni `unstable` e `testing`.

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

In ciascuna delle aree, vi è una cartella per i pacchetti sorgente (`source`) e una cartella per ogni architettura supportata (`binary-i386`, `binary-amd64`, etc.).

L'area `main` contiene altre cartelle che contengono le immagini del disco e alcuni pezzi essenziali della documentazione necessari per l'installazione della distribuzione Debian su una specifica architettura (`disk-i386`, `disk-amd64`, etc.).

4.6.1 Sezioni

La sezione `main` dell'archivio Debian è ciò che costituisce **ufficiale la distribuzione Debian**. La sezione `main` è ufficiale perché soddisfa pienamente tutte le nostre linee guida. Le altre due sezioni no, a gradi diversi; in quanto tali, esse **non** sono ufficialmente parte di Debian.

Ogni pacchetto nella sezione main deve soddisfare le Linee guida Debian per il sofware libero (DFSG) e tutti gli altri requisiti delle policy come descritto nel Debian Policy Manual. Le DFSG sono la nostra definizione del «software libero». Si controlli il Debian Policy Manual per maggiori dettagli.

I pacchetti nella sezione `contrib` sono tenuti a rispettare le DFSG, ma possono non soddisfare altri requisiti. Ad esempio, possono dipendere da pacchetti non-free.

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` sec-

tions, one can avoid any legal risks. Some packages in the `non-free` section do not allow commercial distribution, for example.

D'altra parte, un venditore di CD-ROM potrebbe controllare facilmente le singole licenze dei pacchetti presenti in `non-free` e includere nel CD-ROM il maggior numero consentito. (Dal momento che questo varia da fornitore a fornitore, questo lavoro non può essere fatto dagli sviluppatori 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 Le architetture

I primi tempi, il kernel Linux era disponibile solo per le piattaforme Intel i386 (o superiore), e così è stato per Debian. Ma, man mano che Linux è diventato sempre più popolare, il kernel è stato portato su altre architetture e Debian iniziò a supportarle. E siccome sostenere così tanto hardware non fosse abbastanza, Debian ha deciso di costruire alcuni port sulla base di altri kernel Unix, come `hurd` e `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`).

Informazioni per gli sviluppatori e gli utenti su port specifici sono disponibili presso le pagine web dei Port di Debian.

4.6.3 I pacchetti

Ci sono due tipi di pacchetti Debian, e cioè pacchetti `source (sorgente)` e `binary (binario)`.

A seconda del formato del pacchetto sorgente, esso sarà composto da uno o più file in aggiunta all'obbligatorio `.dsc`:

- con il formato «1.0», esso ha sia un file `.tar.gz` sia un file `.orig.tar.gz` sia un `.diff.gz`;
- con il formato «3.0 (quilt)», esso ha una un archivio tar originale `.orig.tar.{gz,bz2,xz}` obbligatoria, diversi opzionali archivi tar `.orig-`*component*`.tar.{gz,bz2,xz}` originali aggiuntivi e un archivio tar debian obbligatorio `debian.tar.{gz,bz2,xz}`;
- con il formato «3.0 (native)», si ha solo un unico archivio tar `.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.

Il file `.dsc` elenca tutti i file nel pacchetto sorgente insieme ai codici di controllo (`md5sum`) e alcune ulteriori informazioni sul pacchetto (maintainer, versione, etc.).

4.6.4 Le distribuzioni

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`).

Una distribuzione comprende pacchetti binari e sorgenti Debian, e i rispettivi file indice `Sources` e `Packages`, che contengono le informazioni degli header di tutti quei pacchetti. I primi sono tenuti nella cartella `pool/`, mentre i secondi sono tenuti neella cartella `dists/` dell'archivio (per retro-compatibilità).

4.6.4.1 Stable, testing e unstable

Ci sono sempre distribuzioni chiamate `stable` (residente in `dists/stable`), `testing` (residente in `dists/testing`) e `unstable` (residente in `dists/unstable`). Ciò riflette il processo di sviluppo del progetto Debian.

Lo sviluppo attivo è fatto nella distribuzione `unstable` (è per questo che questa distribuzione è a volte chiamata `distribuzione di sviluppo`). Ogni sviluppatore Debian può aggiornare i propri pacchetti in questa distribuzione, in qualsiasi momento. Così, il contenuto di questa distribuzione cambia di giorno in giorno. Dal momento che non viene fatto alcun sforzo particolare per assicurarsi che tutto in questa distribuzione funzioni correttamente, a volte è letteralmente instabile.

La distribuzione *testing* viene generata automaticamente prendendo i pacchetti da `unstable` se soddisfano determinati criteri. Tali criteri dovrebbero assicurare una buona qualità per i pacchetti in `testing`. L'aggiornamento di `testing` è lanciato due volte al giorno, subito dopo che sono stati installati i nuovi pacchetti. Si consulti *La distribuzione testing*.

Dopo un periodo di sviluppo, una volta che il responsabile del rilascio lo ritiene opportuno, la distribuzione `testing` è congelata, il che significa che le politiche che controllano come i pacchetti passano da `unstable` a `testing` sono rese più rigide. I pacchetti che hanno troppi bug vengono rimossi. Non sono consentite modifiche in `testing` tranne che per correzioni di bug. Trascorso un po' di tempo, a seconda dei progressi, la distribuzione `testing` è congelata ancora di più. Dettagli sulla gestione della distribuzione testing sono pubblicati dal team di rilascio su debian-devel-announce. Dopo che i problemi aperti sono stati risolti in modo soddisfacente per il team di rilascio, la distribuzione viene rilasciata. Rilasciare significa che `testing` viene rinominata `stable`, e una nuova copia viene creata per la nuova `testing` e la precedente `stable` viene rinominata `oldstable` e vi rimane fino a quando viene definitivamente archiviata. Una volta archiviata, i contenuti vengono spostati in `archive.debian.org`.

Questo ciclo di sviluppo si basa sul presupposto che la distribuzione `unstable` diventa `stable`, dopo il superamento di un periodo in `testing`. Anche se una distribuzione è considerata stable, alcuni bug inevitabilmente restano: è per questo che la distribuzione stable è aggiornata ogni tanto. Tuttavia, questi aggiornamenti vengono testati molto attentamente e devono essere introdotti nell'archivio singolarmente per ridurre il rischio di introdurre nuovi bug. Si possono trovare le integrazioni proposte alla `stable` nella cartella `proposed-updates`. Questi pacchetti in `proposed-updates` che superano i test sono periodicamente spostati in gruppo nella distribuzione stable e il livello di revisione della distribuzione stable viene incrementato (ad esempio, «6.0» diventa «6.0.1», «5.0.7» diventa «5.0.8», e così via). Fare riferimento a *Caso particolare: caricamenti sulle distribuzioni stable e oldstable* per i dettagli.

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 Maggiori informazioni sulla distribuzione testing

I pacchetti sono generalmente installati nella distribuzione `testing` dopo aver subito un periodo di prova in `unstable`.

Per maggiori dettagli, consultare le informazioni *La distribuzione testing*.

4.6.4.3 Experimental

La distribuzione `experimental` è una distribuzione speciale. Non è una distribuzione completa nello stesso senso di come lo sono `stable`, `testing` e `unstable`. Invece, essa è destinata ad essere una zona di sosta temporanea per software altamente sperimentale dove c'è una buona probabilità che il software potrebbe bloccare il sistema o un software che è semplicemente troppo instabile anche per la distribuzione `unstable` (ma vi è un motivo per pacchettizzarlo comunque). Gli utenti che scaricano e installano i pacchetti da `experimental` si presume siano stati ampiamente avvertiti. In breve, qualsiasi cosa può accadere per la distribuzione `experimental`.

Queste sono le righe di sources.list 5 per `experimental`:

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

Se c'è una possibilità che il software potrebbe fare gravi danni a un sistema, è probabile che sia meglio metterlo in `experimental`. Per esempio, un file system compresso sperimentale dovrebbe probabilmente andare in `experiemental`.

Ogni volta che c'è una nuova versione di un pacchetto che introduce nuove funzionalità, ma rende non funzionanti molte di quelle vecchie, non dovrebbe essere caricato, o essere caricato in `experimental`. Una versione beta nuova di alcuni software che utilizza una configurazione completamente diversa può andare in `experimental`, a discrezione del maintainer. Anche se si sta lavorando su una situazione di aggiornamento incompatibile o complessa, è possibile utilizzare `exerimental` come area di sosta, in modo che i tester possano iniziarci a lavorare prima.

Alcuni software sperimentali possono comunque andare in `unstable`, con alcune avvertenze nella descrizione, ma non è raccomandato perché i pacchetti da `unstable` sono destinati a passare in `testing` e quindi a `stable`. Non si deve aver paura di usare `experimental` dal momento che non causa alcun dolore agli ftpmaster, i pacchetti sperimentali vengono periodicamente rimossi una volta caricato il pacchetto in `unstable` con un numero di versione superiore.

Il nuovo software che non rischia di danneggiare il sistema può andare direttamente in `unstable`.

Un'alternativa a `experimental` è quella di utilizzare il proprio spazio web personale su `people.debian.org`.

4.6.5 Nomi in codice dei rilasci

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.

Dal momento che Debian ha un modello di sviluppo aperto (vale a dire, tutti possono partecipare e seguire lo sviluppo) anche le distribuzioni `unstable` e `testing` sono distribuite su Internet attraverso la rete Debian dei server FTP e HTTP. Così, se avessimo chiamato la cartella che contiene la versione candidata al rilascio `testing`, allora avremmo dovuto rinominarla `stable` quando la versione viene rilasciata, che obbligherebbe tutti i mirror FTP a ri-recuperare l'intera distribuzione (che è piuttosto grande).

D'altra parte, se avessimo chiamato dal principio le cartelle di distribuzione `Debian-x.y`, la gente avrebbe potuto pensare che la versione Debian `x.y` fosse disponibile. (Questo è successo in passato, dove un distributore di CD-ROM Debian 1.0 creò un CD-ROM basato su una versione pre-1.0 di sviluppo. Questa è la ragione per cui il primo rilascio ufficiale di Debian è stato 1.1, e non 1.0.)

Così, i nomi delle cartelle delle distribuzioni presenti nell'archivio sono determinati dai loro nomi in codice e non dal loro stato di rilascio (per esempio, «squeeze»). Questi nomi rimangono invariati durante il periodo di sviluppo e dopo il rilascio; i collegamenti simbolici, che possono essere modificati facilmente, indicano la distribuzione stabile attualmente rilasciata. Ecco perché le cartelle di distribuzione reali utilizzano i `nomi in codice`, mentre i collegamenti simbolici per `stable`, `testing` e `unstable` puntano alle appropriate cartelle di rilascio.

4.7 Mirror di Debian

I vari archivi di download e il sito web hanno diversi mirror disponibili al fine di alleviare i nostri server canonici da carico pesante. In effetti, alcuni dei server canonici non sono pubblici: un primo livello di mirror bilancia, invece, il carico. In questo modo, gli utenti accedono sempre ai mirror e si abituano al loro utilizzo, che consente a Debian di gestire meglio i suoi requisiti di larghezza di banda su più server e reti, e praticamente evita agli utenti di martellare una sola posizione primaria. Si noti che il primo strato di mirror è sempre aggiornato per quanto possibile dal momento che si aggiorna quando innescato dai siti interni (ciò viene detto «push mirroring»).

Tutte le informazioni sui mirror Debian, tra cui un elenco di server pubblici FTP/HTTP disponibili, possono essere trovate su https://www.debian.org/mirror/. Questa utile pagina contiene anche informazioni e strumenti che possono essere utili se si è interessati a realizzare il proprio mirror, sia per l'accesso interno che pubblico.

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 Il sistema Incoming

Il sistema Incoming è responsabile della raccolta dei pacchetti aggiornati e li installa nell'archivio Debian. Si compone di un insieme di cartelle e script che vengono installati su `ftp-master.debian.org`.

I pacchetti sono caricati da tutti i maintainer in una cartella denominata `UploadQueue`. Questa cartella viene analizzata ogni pochi minuti da un demone chiamato `queued`, vengono eseguiti file `*.command` e i file `*.changes` restanti e correttamente firmati vengono spostati insieme ai loro file corrispondenti nella directory `unchecked`. Questa cartella non è visibile alla maggior parte degli sviluppatori, dato che ftp-master è riservato; ma viene sottoposta a scansione ogni 15 minuti dallo script `dak process-upload`, che verifica l'integrità dei pacchetti caricati e le loro firme crittografiche. Se il pacchetto è considerato pronto per essere installato, viene spostato nella cartella `done`. Se questo è il primo caricamento del pacchetto (o ha nuovi pacchetti binari), viene spostato nella cartella `new`, dove attende l'approvazione da parte degli ftpmaster. Se il pacchetto contiene file da installare a mano viene spostato nella cartella `byhand`, dove attende l'installazione manuale degli ftpmasters. Altrimenti, se è stato rilevato qualche errore, il pacchetto viene rifiutato e viene spostato nella cartella `reject`.

Una volta che il pacchetto viene accettato, il sistema invia una email di conferma al maintainer e chiude tutti i bug marcati come corretti dal caricamento, e gli auto-builder possono iniziare a ricompilarlo. Il pacchetto è ora accessibile al pubblico presso https://incoming.debian.org/ fino a quando non sarà realmente installato nell'archivio Debian. Questo accade quattro volte al giorno (ed è chiamata anche la «dinstall run» per ragioni storiche), il pacchetto viene quindi rimosso da incoming ed installato nel pool insieme a tutti gli altri pacchetti. Una volta che tutti gli altri aggiornamenti (che generano i nuovi file di indice `Packages` e `Sources` per esempio) sono stati apportati, uno speciale script è chiamato per chiedere a tutti i mirror primari di aggiornarsi.

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.

Anche se ftp-master è riservato, una copia dell'installazione è disponibile per tutti gli sviluppatori su `mirror.ftp-master.debian.org`.

4.9 Informazioni sul pacchetto

4.9.1 Sul web

Ogni pacchetto ha diverse pagine web dedicate. `http://packages.debian.org/`*nome-pachetto* visualizza ogni versione del pacchetto disponibile nelle varie distribuzioni. Ogni versione punta ad una pagina che fornisce informazioni, compresa la descrizione del pacchetto, le dipendenze, e i collegamenti per il suo scaricamento.

Il sistema di tracciamento dei bug mantiene traccia dei bug di ogni pacchetto. È possibile visualizzare i bug di un determinato pacchetto all'URL `http://bugs.debian.org/`*nome-pacchetto*.

4.9.2 L'utility `dak ls`

`dak ls` è parte della suite di strumenti di dak, ed elenca le versioni dei pacchetti disponibili per tutte le distribuzioni e le architetture conosciute. Lo strumento `dak` è disponibile su `ftp-master.debian.org`, e sul mirror `mirror.ftp-master.debian.org`. Esso utilizza un singolo argomento corrispondente a un nome di un pacchetto. Un esempio servirà a chiarire meglio:

```
$ 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
```

In questo esempio, si può vedere che la versione in `unstable` differisce da quella in `testing` e che c'è stato un NMU solo binario del pacchetto per tutte le architetture. Ciascuna versione del pacchetto è stata ricompilata su tutte le architetture.

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 The Debian Package Tracker

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.

Il PTS ha una interfaccia web all'indirizzo http://packages.qa.debian.org/, che mette insieme molte informazioni su ogni pacchetto sorgente. È dotato di molti collegameti utili (BTS, statistiche QA, informazioni sui contatti, lo stato di traduzione DDTP, log dei buildd) e raccoglie molte informazioni da vari luoghi (le ultime 30 voci del changelog, lo stato di testing, etc.). È uno strumento molto utile se si vuole sapere che cosa sta succedendo ad uno specifico pacchetto sorgente. Inoltre c'è un modulo che permette una facile iscrizione al PTS via email.

È possibile accedere direttamente alla pagina web relativa ad uno specifico pacchetto sorgente con un URL del tipo `http://packages.qa.debian.org/`*pacchettosorgente*.

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 Panoramica dei pacchetti per sviluppatori

Un portale web QA (garanzia di qualità) è disponibile all'indirizzo https://qa.debian.org/developer.php che visualizza una tabella che elenca tutti i pacchetti di un singolo sviluppatore (compresi quelli in cui è elencato come co-maintainer). La tabella fornisce una buona sintesi sui pacchetti dello sviluppatore: numero di bug ordinati per gravità, l'elenco delle versioni disponibili in ogni distribuzione, lo stato di testing e molto altro tra cui collegamenti ad ogni altra informazione utile.

È una buona idea cercare i propri dati regolarmente in modo da non dimenticare eventuali bug aperti e in modo da non dimenticare di quali pacchetti si ha la responsabilità.

4.12 L'installazione FusionForge di Debian: 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.

CAPITOLO 5

Gestione dei pacchetti

Questo capitolo contiene informazioni relative alla creazione, al caricamento, al mantenimento, ed al porting dei pacchetti.

5.1 Nuovi pacchetti

Se si desidera creare un nuovo pacchetto per la distribuzione Debian, si dovrebbe verificare prima l'elenco Work-Needing and Prospective Packages (WNPP). Il controllo dell'elenco WNPP assicura che nessuno stia già lavorando sulla pacchettizzazione del software, e che lo sforzo non sia duplicato. Si leggano le pagine web WNPP per ulteriori informazioni.

Supponendo che nessun altro stia già lavorando sul proprio futuro pacchetto, è necessario poi presentare una segnalazione di bug (*Segnalare i bug*) nei confronti dello pseudo-pacchetto `wnpp` descrivendo come si vuole procedere per creare un nuovo pacchetto, includendo, senza limitarsi ad essa, una descrizione del medesimo, la licenza del futuro pacchetto, e l'URL corrente da dove è possibile scaricarlo.

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.

Inserire una voce `Closes:` *#nnnnn* nel changelog del nuovo pacchetto per la segnalare di bug da chiudere automaticamente una volta che il nuovo pacchetto è installato in archivio (si consulti *Quando i bug vengono chiusi da nuovi upload*).

Se si pensa che il proprio pacchetto abbia bisogno di alcune spiegazioni per gli amministratori della coda NEW dei pacchetti, includerle nel proprio changelog, inviare a `ftpmaster@debian.org` una risposta alla email ricevuta come maintainer dopo il proprio caricamento, o rispondere alla email di rifiuto in caso si stia già effettuando nuovamente il caricamento.

Nel chiudere i bug di sicurezza includete i numeri CVE come `Closes:` *#nnnnn*. Questo è utile al il team di sicurezza per monitorare le vulnerabilità. Se un caricamento è fatto per risolvere il bug prima che l'ID della segnalazione fosse noto, è buona norma aggiornare la voce storica del changelog con il successivo caricamento. Anche in questo caso, Includere tutti i puntatori alle informazioni di fondo nella voce originale del changelog.

Ci sono una serie di motivi per cui chiediamo ai maintainer di dichiarare le loro intenzioni:

- Aiuta il maintainer (potenzialmente nuovo) ad attingere all'esperienza di persone sulla lista, e permette loro di sapere se qualcun altro sta già lavorando su di esso.
- Esso consente ad altre persone di pensare se lavorare sul pacchetto sapendo che c'è già un volontario, così gli sforzi possono essere condivisi.
- Consente al resto dei maintainer di sapere di più sul pacchetto rispetto alla riga di descrizione e alla solita voce del changelog «Initial release» che viene inviata alla `debian-devel-changes@lists.debian.org`.
- È utile per le persone che vivono fuori `unstable` (e che formano la nostra prima linea di tester). Dovremmo incoraggiare queste persone.
- Gli annunci danno ai maintainer e ad altre parti interessate una migliore sensazione di ciò che sta accadendo, e cosa c'è di nuovo, nel progetto.

5.2 Registrare i cambiamenti nel pacchetto

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.

Il `debian/changelog` si adegua ad una certa struttura, con una serie di campi differenti. Un campo di nota, la `distribuzione`, è descritto in *Scegliere una distribuzione*. Maggiori informazioni sulla struttura di questo file si trovano nella sezione della Debian Policy intitolata `debian/changelog`.

Le voci del changelog possono essere usate per chiudere automaticamente i bug Debian quando il pacchetto viene installato nell'archivio. Si consulti *Quando i bug vengono chiusi da nuovi upload*.

È convenzione che la voce del changelog di un pacchetto che contiene una nuova versione originale del software è la seguente:

```
* 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*.

Si consulti anche *Buone pratiche per il debian/changelog*.

5.3 Testare del pacchetto

Prima di caricare il proprio pacchetto, si dovrebbe fare dei test di base su di esso. Come minimo, si dovrebbe provare le seguenti attività (è necessario avere una versione precedente dello stesso pacchetto Debian in giro):

- 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.

  Normalmente, un pacchetto *non* dovrebbe essere caricato se provoca a `lintian` di emettere errori (inizieranno con `E`).

  Per maggiori informazioni su `lintian`, si consulti *lintian*.
- Facoltativamente eseguite `debdiff` (si consulti *debdiff*) per analizzare i cambiamenti da una versione precedente, se ne esiste una.
- 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.
- Rimuovete il pacchetto, quindi reinstallarlo.
- 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*).
- Copiare il pacchetto sorgente in una cartella diversa e provare a spacchettarlo ed a ricompilarlo. Questo testa se il pacchetto si basa su file esistenti al di fuori di esso, o se si basa su permessi che sono stati conservati sui file distribuiti all'interno del `.diff.gz`.

5.4 Struttura del pacchetto sorgente

Ci sono due tipi di pacchetto sorgente Debian:

- i cosiddetti pacchetti `nativi`, dove non c'è distinzione tra i sorgenti originali e le patch applicate per Debian
- i (più comuni) pacchetti dove c'è un file di archivio dei sorgenti originali accompagnato da un altro file che contiene le modifiche apportate da Debian

Per i pacchetti nativi, il pacchetto sorgente include un file di controllo del codice sorgente Debian (`.dsc`) e l'archivio dei sorgenti (`.tar.{gz,bz2,xz}`). Un pacchetto sorgente di un pacchetto non nativo include un file Debian di controllo sorgenti, l'archivio dei sorgenti originali (`.orig.tar.{gz,bz2,xz}`) e le modifiche Debian (`.diff.gz` per il formato sorgente «1.0» o `.debian.tar.{gz,bz2,xz}` per il formato sorgente «3.0 (quilt)»).

Con il formato dei sorgenti «1.0», se un pacchetto è nativo o non è stato determinato da `dpkg-source` al momento della compilazione. Al giorno d'oggi, si raccomanda di essere espliciti sul formato sorgente desiderato mettendo entrambi «3.0 (quilt)» o «3.0 (nativo)» in `debian/source/format`. Il resto di questa sezione riguarda solo i pacchetti non-nativi.

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.

Per impostazione predefinita, `dpkg-genchanges` e `dpkg-buildpackage` includerà il file tar dei sorgenti originali, se e solo se l'attuale voce del changelog ha una versione originale diversa dalla voce precedente. Questo comportamento può essere modificato utilizzando `-sa` per includere sempre o `-sd` per lasciarlo sempre fuori.

Se nessun sorgente originale è incluso nel caricamento, il file tar dei sorgenti originali utilizzato da `dpkg-source` quando fu costruito il file `.dsc` e diff da caricare *deve* essere del tutto identico a quello già presente in archivio.

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 Scegliere una distribuzione

Ogni caricamento deve specificare a quale distribuzione il pacchetto è destinato. Il processo di compilazione del pacchetto estrae questa informazione dalla prima riga del `debian/changelog` e la inserisce nel campo `Distribution` del file `.changes`.

I pacchetti sono generalmente caricati in `unstable`. I caricamenti in `unstable` o `experimental` dovrebbero utilizzare questi nomi nelle voci del changelog; caricamenti per altre suite dovrebbero utlizzare il nome della suite, in modo da evitare ambiguità.

In realtà, ci sono altre due possibili distribuzioni: `stable-security` e `testing-security`, ma si legga *Gestione di bug relativi alla sicurezza* per ulteriori informazioni su queste ultime.

Non è possibile caricare un pacchetto in più distribuzioni contemporaneamente.

5.5.1 Caso particolare: caricamenti sulle distribuzioni `stable` e `oldstable`

Uploading to `stable` means that the package will be transferred to the `proposed-updates-new` queue for review by the stable release managers, and if approved will be installed in the `stable-proposed-updates` directory of the Debian archive. From there, it will be included in `stable` with the next point release.

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.

caricamenti su distribuzioni `oldstable` sono possibili a patto che non siano state archiviate. Valgono le stesse regole per `stable`.

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 *Gestione di bug relativi alla sicurezza* 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 *Caso particolare: caricamenti sulle distribuzioni stable e oldstable*. 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 Caso particolare: caricamenti su `testing/testing-proposed-updates`

Consultare le informazioni nella *Aggiornamenti diretti di testing* per i dettagli.

5.6 Caricare un pacchetto

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 *Nuovi pacchetti*) 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 *Marcare i pacchetti non-free come compilabili automaticamente*).

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 Caricamento su `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).

Notare che è necessario trasferire il file delle modifiche alla fine. In caso contrario, il caricamento potrebbe essere respinto in quanto il software di mantenimento analizzerà il file delle modifiche e noterà che non tutti i file sono stati caricati.

È inoltre possibile trovare i pacchetti Debian *dupload* o *dput* utili quando si caricano i pacchetti. Questi comodi programmi aiutano ad automatizzare il processo di caricamento dei pacchetti in 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 *The Debian Package Tracker* for transition warnings to avoid making uploads that disrupt ongoing transitions.

5.6.3 Caricamenti differiti

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 *Caricamenti dei Non-Maintainer (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.

Un caricamento nella cartella differita fa mantenere il pacchetto nella coda caricamenti differiti. Quando il tempo di attesa specificato è terminato, il pacchetto viene spostato nella cartella regolare incoming per l'elaborazione. Questo viene fatto attraverso il caricamento di `ftp.upload.debian.org` nella cartella di caricamento`DELAYED/`*X*`-day` (*X* tra 0 e 15). 0-day viene caricato più volte al giorno in `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 Caricamenti di sicurezza

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 *Gestione di bug relativi alla sicurezza*.

5.6.5 Altre code di caricamento

Vi è una coda di caricamento alternativa in Europa a ftp://ftp.eu.upload.debian.org/pub/UploadQueue/. Funziona allo stesso modo come `ftp.upload.debian.org`, ma dovrebbe essere più veloce per gli sviluppatori europei.

I pacchetti possono anche essere caricati via ssh al `ssh.upload.debian.org`; i file devono essere messi in `/srv/upload.debian.org/UploadQueue`. Questa coda non supporta *Caricamenti differiti*.

5.6.6 Notifications

I maintainer dell'archivio Debian sono responsabili per la gestione dei caricamenti dei pacchetti. Per la maggior parte, i caricamenti sono gestiti automaticamente su base giornaliera dagli strumenti di manutenzione, `dak process-upload`. In particolare, aggiornamenti di pacchetti esistenti nella distribuzione `unstable` sono gestiti automaticamente. In altri casi, in particolare per i nuovi pacchetti, il posizionamento del pacchetto caricato nella distribuzione viene gestita manualmente. Quando i caricamenti sono gestiti manualmente, la modifica all'archivio può richiedere un certo tempo. Si sia pazienti.

In ogni caso, si riceverà una email di notifica per indicare che il pacchetto è stato aggiunto all'archivio, inoltre indica quali bug saranno chiusi dal caricamento. Esaminare attentamente questa notifica, controllando se qualche bug che si intendeva chiudere è stato tralasciato.

La notifica di installazione include anche informazioni sulla sezione nella quale il pacchetto è stato inserito. Se vi è una disparità, riceverai una email separata di notifica che te lo comunicherà. Si continui a leggere di seguito.

Si noti che se si carica tramite le code, il software demone delle code invierà anche una notifica via email.

Also note that new uploads are announced on the *Canali 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 Specificare la sezione del pacchetto, sottosezione e la priorità

I campi `Section` e `Priority` del file `debian/control` in realtà non specificano dove il file verrà inserito nell'archivio, né la sua priorità. Al fine di mantenere l'integrità complessiva dell'archivio, sono i maintainer dell'archivio che hanno il controllo su questi campi. I valori del file `debian/control` sono in realtà solo suggerimenti.

I maintainer dell'archivio mantengono traccia delle sezioni e delle priorità canoniche per i pacchetti presenti nel file `override`. Se c'è una disparità tra il file `override` e campi del pacchetto come indicato in `debian/control`, allora si riceverà una email che sottolinerà la divergenza nel momento in cui il pacchetto viene installato nell'archivio. È possibile correggere il file`debian/control` per il successivo caricamento, oppure si potrebbe desiderare di fare un cambiamento nel `file di override`.

Per modificare la sezione attuale nella quale il pacchetto è stato inserito, è necessario prima assicurarsi che il `debian/control` nel pacchetto sia preciso. Successivamente, si crei un bug su `ftp.debian.org` chiedendo che la sezione o la priorità per il pacchetto siano modificati dalla vecchia sezione o priorità a quella nuova. Si utilizzi un Subject del tipo `override: Package1: sezione/priorità, [...], PACKAGEX: sezione/priorità`, e includere la motivazione per la modifica nel corpo della segnalazione del bug.

Per ulteriori informazioni sugli `file di override`, si consulti dpkg-scanpackages 1 e https://www.debian.org/Bugs/Developer#maintincorrect.

Si noti che il campo `Section` descrive sia la sezione che la sottosezione, che sono descritti nella *Sezioni*. Se la sezione è la principale, dovrebbe essere omessa. L'elenco delle sottosezioni ammissibili può essere trovato in 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 *Testare del pacchetto*.

5.9 Gestione dei bug

Ogni sviluppatore deve essere capace di lavorare con il Debian bug tracking system. Questo implica la conoscenza di come creare propriamente le segnalazioni di bug (si consulti *Segnalare i bug*), di come modificarli e riordinarli, e di come processarli e chiuderli.

Le caratteristiche del sistema di tracciamento dei bug sono descritte nella documentazione BTS per gli sviluppatori. Questo include la chiusura bug, l'invio di messaggi di riepilogo, l'assegnazione dei livelli di gravità e tag, il marcare i bug come inoltrati, e altre questioni.

Operazioni quali la riassegnazione di bug ad altri pacchetti, fondendo le segnalazioni di bug separate sullo stesso problema, o la riapertura di bug quando sono chiusi prematuramente, vengono gestite utilizzando il cosiddetto server di controllo di posta elettronica. Tutti i comandi disponibili su questo server sono descritti nella documentazione del server di controllo BTS.

5.9.1 Monitoraggio dei bug

Se si vuole essere un buon maintainer, si dovrebbe verificare periodicamente il Debian bug tracking system (BTS) per i propri pacchetti. Il BTS contiene tutti i bug aperti per i propri pacchetti. È possibile controllarli sfogliando questa pagina: `http://bugs.debian.org/`*yourlogin*`@debian.org`.

I maintainer interagiscono con le BTS attraverso indirizzi di posta elettronica a `bugs.debian.org`. La documentazione sui comandi disponibili può essere trovata su https://www.debian.org/Bugs/, oppure, se è stato installato il pacchetto `doc-debian`, è possibile guardare i file locali `/usr/share/doc/debian/bug-*`.

Alcuni trovano utile avere rapporti periodici sul bug aperti. È possibile aggiungere un processo di cron come segue, se si vuole ottenere una email settimanale che illustra tutti i bug aperti per i propri pacchetti:

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

Sostituite *address* con il proprio indirizzo ufficiale di maintainer Debian.

5.9.2 Rispondere ai bug

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.

Se si ottiene un bug che cita FTBFS, questo significa non si riesce a compilare dai sorgenti. Gli autori dei port utilizzano spesso questo acronimo.

Una volta che si è affrontata una segnalazione di bug (e.g. correggendolo), lo si contrassegni come `done` (lo si chiuda) con l'invio di un messaggio di spiegazione a *123*`-done@bugs.debian.org`. Se si sta correggendo un bug modificando e caricando il pacchetto, è possibile automatizzare la chiusura del bug come descritto in *Quando i bug vengono chiusi da nuovi upload*.

Mai si dovrebbe chiudere un bug tramite il comando del bug server `close` inviato a `control@bugs.debian.org`. Se lo fate, il mittente originale non riceverà alcuna informazione sul perché il bug è stato chiuso.

5.9.3 Pulizia dei bug

Come maintainer del pacchetto, spesso si trovano bug in altri pacchetti o si hanno bug segnalati sui propri pacchetti ma che in realtà sono bug di altri pacchetti. Le caratteristiche del sistema di tracciamento dei bug sono descritte nella documentazione BTS per gli sviluppatori Debian. Operazioni come la riassegnazione, l'unione e segnalazioni di bug, l'etichettatura sono descritte nella BTS control server documentation. Questa sezione contiene alcune linee guida per la gestione dei propri bug, sulla base dell'esperienza collettiva degli sviluppatori Debian.

Segnalare bug per problemi che si trovano in altri pacchetti è uno dei doveri civici di maintainer, si consulti *Segnalare i bug* per i dettagli. Tuttavia, la gestione dei bug nei propri pacchetti è ancora più importante.

Ecco un elenco di passaggi che si possono seguire per gestire una segnalazione di errore:

1. Decidere se la segnalazione corrisponde ad un vero e proprio bug o meno. A volte gli utenti invocano un programma nel modo sbagliato perché non hanno letto la documentazione. Se la diagnosi è questa, basta chiudere il bug con informazioni sufficienti per consentire agli utenti di correggere il loro problema (dare indicazioni sulla buona documentazione e così via). Se la stessa segnalazione viene presentata più e più volte ci si potrebbe chiedere se la documentazione sia sufficiente o se il programma non rilevi il suo cattivo uso al fine di fornire un messaggio di errore. Questo è un problema che può aver bisogno di essere risolto con l'autore originale.

   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. Se il bug è reale ma è causato da un altro pacchetto, basta riassegnare il bug al pacchetto giusto. Se non si sa a quale pacchetto dovrebbe essere riassegnato, si dovrebbe chiedere aiuto su *Canali IRC* o su `debian-devel@lists.debian.org`. Informare il maintainer del pacchetto al quale si riassegna il bug, per esempio mettendo in Cc: al messaggio che fa la riassegnazione a *packagename*`@packages.debian.org` e spiegando le proprie motivazioni nel corpo della email. Si noti che una semplice riassegnazione *non* è inviata ai maintainer del pacchetto al quale viene riassegnato, quindi non avranno modo di saperlo fino a quando consulteranno la panoramica dei bug per i loro pacchetti.

   Se il bug interessa il funzionamento del proprio pacchetto, si consideri di clonare il bug e di riassegnarlo al pacchetto che realmente provoca il comportamento. In caso contrario, il bug non verrà mostrato nella lista dei bug del proprio pacchetto, inducendo gli utenti a segnalare lo stesso problema più e più volte. Si dovrebbe bloccare «il proprio» bug con il bug riassegnato, clonato per documentare la relazione.

3. A volte è necessario anche per regolare la gravità del bug in modo che corrisponda alla propria definizione di gravità. Questo perché le persone tendono a gonfiare la gravità dei bug per assicurarsi che i loro bug siano risolti rapidamente. Alcuni bug possono anche essere lasciati con gravità wishlist quando il cambiamento richiesto è solo estetico.

4. Se il bug è reale, ma lo stesso problema è già stato segnalato da qualcun altro, allora le due segnalazioni di bug rilevanti dovrebbero essere fuse in una sola utilizzando il comando merge del BTS. In questo modo, quando il bug viene corretto, tutti i mittenti ne saranno informati. (Si noti, tuttavia, che i messaggi di posta elettronica inviati al mittente di un solo bug non saranno automaticamente inviati al mittente dell'altra segnalazione.) Per maggiori dettagli sui tecnicismi del comando merge e simili, il comando unmerge, si consulti la documentazione del server di controllo BTS.

5. Il mittente del bug potrebbe aver dimenticato di fornire alcune informazioni, nel qual caso si deve chiedergli le informazioni necessarie. A tal proposito è possibile marcare il bug con il tag `moreinfo`. Inoltre se non è possibile riprodurre il bug, lo si etichetta come `unreproducible`. Chiunque può riprodurre il bug è allora invitato a fornire ulteriori informazioni su come riprodurlo. Dopo pochi mesi, se queste informazioni non sono state inviate da nessuno, il bug può essere chiuso.

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 *Coordinamento con gli sviluppatori originali* 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. Se è stato sistemato un errore nella copia locale, o se una correzione è stata committata nel repository VCS, è possibile identificare il bug come `pending`, per far sapere che il bug è stato corretto e che sarà chiuso con il prossimo caricamento (si aggiunga `closes:` nel `changelog`). Questo è particolarmente utile se si è in diversi sviluppatori che lavorano sullo stesso pacchetto.

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 *Quando i bug vengono chiusi da nuovi upload*.

5.9.4 Quando i bug vengono chiusi da nuovi upload

Cosi come bug e problemi sono corretti nei propri pacchetti, è responsabilità del maintainer del pacchetto chiudere questi bug. Tuttavia, non è necessario chiudere un bug fino a quando il pacchetto che corregge il bug è stato accettato nell'archivio Debian. Pertanto, una volta che si ottiene la notifica che il proprio pacchetto aggiornato è stato installato nell'archivio, si può e si deve chiudere il bug nel BTS. Inoltre, il bug dovrebbe essere chiuso con la versione corretta.

Tuttavia, è possibile evitare di dover chiudere manualmente i bug dopo il caricamento - basta elencare i bug corretti nel proprio file `debian/changelog`, seguendo una certa sintassi, e il software di manutenzione chiuderà il bug. Per esempio:

```
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.
```

Tecnicamente parlando, la seguente espressione regolare Perl descrive come i changelog che chiudono dei bug sono identificati:

```
/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).

Storicamente, i caricamenti individuati come *Caricamenti dei Non-Maintainer (NMU)* sono stati contrassegnati come `fixed` invece di essere chiusi, ma questa pratica è stata cessata con l'avvento del versionamento. Lo stesso vale per il tag `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.

Tenete a mente che non è obbligatorio chiudere i bug utilizzando il changelog come descritto sopra. Se si vuole semplicemente chiudere bug che non hanno nulla a che vedere con un caricamento che si è fatto, lo si faccia inviando tramite email una spiegazione a *XXX*`-done@bugs.debian.org`. **Non** chiudete bug nella voce del changelog di una versione se le modifiche in quella versione del pacchetto non hanno alcuna attinenza con il bug.

Per informazioni generali su come scrivere i propri contenuti di changelog, si consulti *Buone pratiche per il debian/changelog*.

5.9.5 Gestione di bug relativi alla sicurezza

A causa della loro natura sensibile, i bug relativi alla sicurezza devono essere maneggiati con cura. Esiste Debian Security Team per coordinare questa attività, tenendo traccia dei problemi di sicurezza in sospeso, aiutando i maintainer con problemi di sicurezza o sistemandoli loro stessi, inviando avvisi di sicurezza, e mantenendo `security.debian.org`.

Quando si viene a conoscenza di un bug relativo alla sicurezza in un pacchetto Debian, anche se non si è il maintainer, si raccolga informazioni pertinenti in merito al problema, e si contatti immediatamente il team di sicurezza, preferibilmente presentando il ticket nella propria Request Tracker. Si consulti http://wiki.debian.org/rt.debian.org#Security_Team. In alternativa si può mandare una email `team@security.debian.org`. **NON FARE L'UPLOAD** di pacchetti per `stable` senza contattare il team. Le informazioni utili comprendono, per esempio:

- Se il bug è già di dominio pubblico.
- Quali versioni del pacchetto sono note per essere interessate dal bug. Si controlli ogni versione che è presente in un rilascio supportato di Debian, così come `testing` e `unstable`.
- La natura della correzione, se qualcuna è disponibile (patch sono particolarmente utili)
- Any fixed packages that you have prepared yourself (send the resulting debdiff or alternatively only the `.diff.gz` and `.dsc` files and read *Preparazione di pacchetti per indirizzare i problemi di sicurezza* first)
- Qualsiasi tipo di assistenza si è in grado di fornire per aiutare con i test (exploit, test di regressione, etc.)
- Tutte le informazioni necessarie per la consulenza (si consulti *Avvisi di sicurezza*)

Come maintainer del pacchetto, si ha la responsabilità di mantenerlo, anche nella versione stabile. Si è nella posizione migliore per valutare le patch e testare i pacchetti aggiornati, quindi consultare le sezioni di seguito su come preparare i pacchetti per il Security Team.

5.9.5.1 Debian Security Tracker

Il team di sicurezza mantiene una banca dati centrale, il Debian Security Tracker. Questa contiene tutte le informazioni pubbliche che sono note sui problemi di sicurezza: quali pacchetti e versioni sono interessate o corrette, e quindi se stable, testing e/o unstable sono vulnerabili. Informazioni che sono ancora confidenziali non vengono aggiunte al tracker.

È possibile cercare per un problema specifico, ma anche sul nome del pacchetto. Cercare il proprio pacchetto per vedere quali problemi sono ancora aperti. Se è possibile, fornire ulteriori informazioni su questi problemi, o di aiutare ad indirizzarli nel proprio pacchetto. Le istruzioni si trovano sulle pagine web del tracker.

5.9.5.2 Riservatezza

A differenza di molte altre attività all'interno di Debian, le informazioni su problemi di sicurezza devono talvolta essere mantenute private per un certo tempo. Questo consente ai distributori di software di coordinare la loro divulgazione al fine di ridurre al minimo l'esposizione dei loro utenti. Se questo è il caso dipende dalla natura del problema e dalla correzione corrispondente, e se è già di dominio pubblico.

Ci sono diversi modi con cui gli sviluppatori possono venire a conoscenza di problemi di sicurezza:

- è stato notato su un forum pubblico (mailing list, sito web, etc.)
- qualcuno deposita una segnalazione di bug
- qualcuno li informa via email privata

Nei primi due casi, l'informazione è pubblica ed è importante avere una correzione il più presto possibile. Nell'ultimo caso, tuttavia, potrebbe non essere una informazione pubblica. In questo caso ci sono alcune possibili opzioni per affrontare il problema:

- Se l'esposizione di sicurezza è minore, talvolta non è necessario mantenere il problema un segreto e una correzione devrebbe essere fatta e rilasciata.
- Se il problema è grave, è preferibile condividere le informazioni con altri fornitori e coordinare un rilascio. Il team di sicurezza si mantiene in contatto con le varie organizzazioni e gli individui e può prendersi cura di questo.

In tutti i casi, se la persona che segnala il problema chiede di non divulgarlo, tali richieste devono essere onorate, con l'ovvia eccezione di informare il team di sicurezza in modo che una soluzione possa essere prodotta per un rilascio stabile di Debian. Quando si inviano informazioni riservate al team di sicurezza, ci si assicuri di indicare questo fatto.

Si tenga presente che se è necessaria la segretezza non si può caricare una correzione in `unstable` (o in qualsiasi altro luogo, come un repository pubblico VCS). Non è sufficiente offuscare i dettagli della modifica, dato che il codice stesso è pubblico, e può (e sarà) esaminato dal pubblico in generale.

Ci sono due motivi per il rilascio di informazioni anche se è richiesto il segreto: il problema è conosciuto da un po', o il problema o l'exploit è diventato pubblico.

Il team di sicurezza ha un PGP-key per abilitare la comunicazione crittografata su questioni delicate. Si consulti la Security Team FAQ per i dettagli.

5.9.5.3 Avvisi di sicurezza

Gli avvisi di sicurezza vengono rilasciati solo per la corrente, rilasciata distribuzione stabile, e *non* per `testing` o `unstable`. Quando viene rilasciata, gli avvisi vengono inviati alla mailing list `debian-security-announce@lists.debian.org` e pubblicate sulla pagina web di sicurezza. Gli avvisi di sicurezza sono scritti e pubblicati dal team di sicurezza. Ma di certo non ci restano male se un maintainer è in grado di fornirgli alcune delle informazioni, o scrivere una parte del testo. Le informazioni che devono essere presenti in un avviso comprendono:

- Una descrizione del problema e il suo campo di applicazione, tra cui:
  - Il tipo di problema (l'escalation dei privilegi, denial of service, etc.)
  - Quali privilegi possono essere acquisiti, e da chi (se alcuni)
  - Come può essere sfruttata
  - Se è sfruttabile da remoto o in locale
  - Come è stato risolto il problema

  Queste informazioni consentono agli utenti di valutare la minaccia per i loro sistemi.
- Numeri di versione dei pacchetti coinvolti
- Numeri di versione dei pacchetti corretti
- Informazioni su dove ottenere i pacchetti aggiornati (di solito dall'archivio di sicurezza di Debian)
- References to upstream advisories, CVE identifiers, and any other information useful in cross-referencing the vulnerability

5.9.5.4 Preparazione di pacchetti per indirizzare i problemi di sicurezza

Un modo con cui si può aiutare il team di sicurezza nei suoi compiti è quello di fornirgli pacchetti corretti adatti per un avviso di sicurezza per il rilascio stabile di 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.

Ciò significa che il passaggio a una nuova versione originale non è una buona soluzione. Invece, le rilevanti modifiche devono essere adattate alla versione presente nella attuale rilascio stabile di Debian. In generale, i maintainer originali sono disposti ad aiutare se necessario. In caso contrario, il team di sicurezza di Debian può essere in grado di aiutare.

In alcuni casi, non è possibile adattare una correzione di sicurezza, per esempio quando grandi quantità di codice sorgente dovrebbero essere modificate o riscritte. Se questo accade, può essere necessario passare ad una nuova versione. Tuttavia, questo è fatto solo in situazioni estreme, e ci si deve sempre coordinare che con il team della sicurezza.

A questo si collega un'altra importante linea guida: verificare sempre le modifiche. Se si dispone di un exploit, provare e vedere se davvero ha avuto successo sul pacchetto senza patch e fallisce sul pacchetto corretto. Provare altre, anche normali azioni, dato che a volte una correzione di sicurezza può rompere in modi sottili caratteristiche apparentemente non correlate.

NON includere eventuali modifiche nel proprio pacchetto che non sono direttamente collegate alla correzione della vulnerabilità. Queste avranno bisogno solo di essere annullate, e questo richiede tempo. Se ci sono altri bug nel pacchetto che si desidera correggere, si faccia un caricamento su proposed-updates nel solito modo, dopo che l'avviso di sicurezza viene pubblicato. Il meccanismo di aggiornamento della sicurezza non è un mezzo per introdurre modifiche al pacchetto che altrimenti verrebbero respinte per il rilascio stabile, quindi per favore non si tenti di farlo.

Si verifichino e testino le modifiche per quanto possibile. Si controllino ripetutamente le differenze rispetto alla versione precedente (`interdiff` dal pacchetto `patchutils` e `debdiff` da `devscripts` sono strumenti utili per questo, si consulti *debdiff*).

Assicurarsi di verificare i seguenti elementi:

- **Individuare la giusta distribuzione** nel proprio `debian/changelog`. Per `stable` questa è `stable-security` e per `testing` questa è `testing-security`, e per la precedente versione stable, questa è `oldstable-security`. Non si punti a *distribution*`-proposed-updates` o `stable`!
- Si creino voci descrittive e significative nel changelog. Altri faranno affidamento su di loro per determinare se un particolare bug è stato risolto. Si aggiungano istruzioni `closes:` per eventuali **bug di Debian** pubblicati. Sempre includete un riferimento esterno, preferibilmente un **identificatore CVE**, in modo che ci possa essere un riferimento incrociato. Tuttavia, se un identificatore CVE non è ancora stato assegnato, non attenderlo ma continuare il processo. L'identificatore può essere referenziato più tardi.
- Assicurarsi che il **version number** sia corretto. Esso deve essere maggiore del pacchetto corrente, ma minore delle versioni del pacchetto in distribuzioni successive. In caso di dubbio, provare con `dpkg --compare-versions`. Fare attenzione a non riutilizzare un numero di versione che è già stato utilizzato per un caricamento precedente, o uno che è in conflitto con un binNMU. La convenzione è quella di aggiungere *+codename* `1`, ad esempio, `1:2.4.3-4+lenny1`, ovviamente aumentando di 1 ad ogni aggiornamento successivo.
- A meno che il sorgente originale sia stato prima caricato su `security.debian.org` (grazie ad un aggiornamento di sicurezza precedente), costruite il file da caricare **con tutto il codice originale** (`dpkg-buildpackage -sa`). Se vi è stato un precedente caricamento su `security.debian.org` con la stessa versione dell'originale, si può caricare senza sorgenti originali (`dpkg-buildpackage -sd`).
- Assicurarsi di utilizzare **esattamente lo stesso``*.orig.tar.{gz,bz2,xz}``** come usato nell'archivio normale, altrimenti non è possibile spostare la correzione di sicurezza negli archivi principali dopo.
- Si compili il pacchetto su un **sistema pulito**, che ha installati solo i pacchetti della distribuzione per la quale si sta costruendo. Se non si dispone di un tale sistema, è possibile utilizzare una macchina di debian.org (si consulti *Le macchine Debian*) oppure si configuri un chroot (si consulti *pbuilder* e *debootstrap*).

5.9.5.5 Caricamento del pacchetto corretto

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.

NON caricare la correzione su `proposed-updates`, senza coordinarsi con il team di sicurezza. I pacchetti da `security.debian.org` saranno copiati nella cartella `proposed-updates` automaticamente. Se un pacchetto con lo stesso numero di versione o uno più alto è già installato nell'archivio, l'aggiornamento per la sicurezza sarà rifiutato dal sistema di archiviazione. In questo modo, la distribuzione stabile non avrà un aggiornamento di sicurezza per questo pacchetto.

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/`.

Una volta che un caricamento nella coda di sicurezza è stato accettato, il pacchetto viene automaticamente compilato per tutte le architetture e conservato per la verifica da parte del team di sicurezza.

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.

Se un membro del team di sicurezza accetta un pacchetto, verrà installato su `security.debian.org`, così come proposto per la corretta *distribution*`-proposed-updates` in `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 Lo spostamento, la rimozione, la ridenominazione, l'adozione, e rendere orfani i pacchetti

Alcune operazioni di manipolazione di archivi non sono automatizzate nel processo di caricamento di Debian. Queste procedure devono essere seguite manualmente dai maintainer. Questo capitolo fornisce le linee guida su cosa fare in questi casi.

5.11.1 Spostare i pacchetti

A volte un pacchetto cambierà la sua sezione. Per esempio, un pacchetto dalla sezione `non-free` potrebbe essere GPLizzato in una versione successiva, nel qual caso il pacchetto dovrebbe essere spostato in «main» o «contrib».[1]

Se è necessario modificare la sezione per uno dei propri pacchetti, si modifichino le informazioni di controllo del pacchetto per far posizionare il pacchetto nella sezione desiderata, e caricare nuovamente il pacchetto (si consulti il Debian Policy Manual per i dettagli). È necessario assicurarsi di includere il `.orig.tar.{gz,bz2,xz}` nel proprio caricamento (anche se non si sta caricando una nuova versione), oppure non comparirà nella nuova sezione insieme al resto del pacchetto. Se la nuova sezione è valida, verrà spostata automaticamente. Se non lo è, allora si contatti gli ftpmasters per capire cosa è successo.

Se, d'altra parte, è necessario modificare la `subsection` di uno dei pacchetti (per esempio, «devel», «admin»), la procedura è leggermente diversa. Correggete la subsection come si trova nel file di controllo del pacchetto, e caricatelo nuovamente. Inoltre, è necessario per avere il file di override aggiornato, come descritto in *Specificare la sezione del pacchetto, sottosezione e la priorità*.

5.11.2 Rimozione dei pacchetti

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

[1] Si consulti il Debian Policy Manual per le linee guida su quale sezione appartiene un pacchetto.

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 *Rimozioni da testing*.)

C'è una sola eccezione quando una richiesta di rimozione esplicita non è necessaria: se un pacchetto (sorgente o binario) non è più compilato dal sorgente, verrà rimosso in modo semiautomatico. Per un pacchetto binario, questo significa che se non vi è più alcun pacchetto sorgente che produce questo pacchetto binario; se il pacchetto binario non è più prodotto per alcune architetture, una richiesta di rimozione è ancora necessaria. Per un pacchetto sorgente, questo significa che tutti i pacchetti binari che a lui si riferiscono devono riferirsi ad un altro pacchetto sorgente.

Nella vostra richiesta di rimozione, è necessario dettagliare le ragioni che giustificano la richiesta. Questo per evitare rimozioni indesiderate e per tenere traccia del perché un pacchetto è stato rimosso. Ad esempio, è possibile specificare il nome del pacchetto che sostituisce quello che deve essere rimosso.

Di solito si chiede solo per la rimozione di un pacchetto che si sta mentenendo. Se si desidera rimuovere un altro pacchetto, è necessario ottenere l'approvazione del suo maintainer. Se il pacchetto resta orfano e quindi non ha un maintainer, si deve prima discutere la richiesta di rimozione su `debian-qa@lists.debian.org`. Se c'è un consenso sul fatto che il pacchetto debba essere rimosso, è necessario riassegnare e rinominare la segnalazione del bug `O:` presentato sul pacchetto `wnpp` invece di presentare un nuovo bug come richiesta di rimozione.

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`.

Una volta che il pacchetto è stato rimosso, i bug del pacchetto devono essere gestiti. Essi dovrebbero essere riassegnati ad un altro pacchetto nel caso in cui il codice vero e proprio si sia evoluto in un altro pacchetto (ad esempio `libfoo12` è stato rimosso perché `libfoo13` lo sostituisce) o chiuso, se il software semplicemente non fa più parte di Debian. Quando si chiude il bug, per evitare di segnare i bug corretti in versioni di pacchetti di rilasci precedenti di Debian, dovrebbero essere contrassegnati come corretti nella versione `<most-recent-version-ever-in-Debian>+rm`.

5.11.2.1 Rimozione di pacchetti da `Incoming`

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 La sostituzione o la ridenominazione dei pacchetti

Quando i maintainer originali a causa di uno dei propri pacchetti hanno scelto di rinominare il loro software (o si è commesso un errore nel nominare il proprio pacchetto), si dovrebbe seguire un processo in due fasi per rinominarlo. Nella prima fase, modificare il file `debian/control` affinché rifletta il nuovo nome e per sostituire, prevedete e risolvete eventuali conflitti con il nome del pacchetto obsoleto (si consulti Debian Policy Manual per i dettagli). Si tenga presente che si deve solo aggiungere una relazione `Provides` se tutti i pacchetti dipendenti dall'obsoleto nome del pacchetto continuano a funzionare dopo la ridenominazione. Una volta caricato il pacchetto e il pacchetto è spostato nell'archivio, si apra un bug nei confronti di `ftp.debian.org` chiedendo di rimuovere il pacchetto con il nome obsoleto (si veda *Rimozione dei pacchetti*). Non si dimentichi allo stesso tempo di riassegnare correttamente i bug del pacchetto.

Altre volte, si può fare un errore nella compilazione del proprio pacchetto e si vuole sostituirlo. L'unico modo per farlo è aumentare il numero di versione e caricarlo. La vecchia versione scadrà nel modo consueto. Si noti che questo vale per ogni parte del proprio pacchetto, compresi i sorgenti: se si desidera sostituire l'archivio dei sorgenti originali del proprio pacchetto, è necessario caricarlo con una versione diversa. Un modo semplice è quello

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

di sostituire `foo_1.00.orig.tar.gz` con `foo_1.00+0.orig.tar.gz` o `foo_1.00.orig.tar.bz2`. Questa restrizione dà ad ogni file sul sito FTP un nome unico, che contribuisce a garantire la coerenza tra i mirror.

5.11.4 Pacchetto orfano

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`.

Maggiori informazioni si possono trovare nelle pagine web di WNPP.

5.11.5 L'adozione di un pacchetto

Un elenco di pacchetti che necessitano di un nuovo maintainer è disponibile nella Work-Needing and Prospective Packages list(WNPP). Se si desidera prendere in consegna la manutenzione di uno qualsiasi dei pacchetti elencati nella WNPP, si prega di dare un'occhiata alla pagina di cui sopra per informazioni e procedure.

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 *Rapportarsi con maintainer non attivi e/o non raggiungibili*.

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).

Se si rileva un vecchio pacchetto, probabilmente si vuole essere elencati come maintainer ufficiale del pacchetto nel sistema di tracciamento dei bug. Questo avverrà automaticamente una volta che si carica una nuova versione con un campo `Maintainer` aggiornato, anche se può richiedere alcune ore dopo che il caricamento sia stato fatto. Se non si prevede di caricare una nuova versione per un po', è possibile utilizzare *The Debian Package Tracker* per ottenere le segnalazioni di bug. Tuttavia, ci si assicuri che il vecchio maintainer non abbia alcun problema con il fatto che in quel periodo continuerà a ricevere le segnalazioni dei bug.

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 Reintrodurre pacchetti

I pacchetti sono spesso rimossi a causa di bug critici, manutentori assenti, troppo pochi utenti o di scarsa qualità in generale. Mentre il processo di reintroduzione è simile al processo di pacchettizzazione iniziale, è possibile evitare alcune insidie facendo prima qualche ricerca storica.

Si dovrebbe verificare il motivo per cui il pacchetto è stato rimosso, in primo luogo. Queste informazioni si possono trovare nella voce rimozione nella sezione news della pagina PTS per il pacchetto o sfogliando il registro dirimossi. Il bug rimozione vi dirà il motivo per cui il pacchetto è stato rimosso e darà qualche indicazione di ciò che è necessario correggere al fine di reintrodurre il pacchetto. Può indicare che il modo migliore di procedere è quello di passare a qualche altro pezzo di software, invece di reintrodurre il pacchetto.

Può essere opportuno contattare gli ex manutentori per scoprire se si sta lavorando per reintrodurre il pacchetto, interessati a co-mantenimento del pacchetto o interessati a sponsorizzare il pacchetto, se necessario.

Si dovrebbe fare tutto ciò di necessario prima di introdurre nuovi pacchetti (*Nuovi pacchetti*).

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.

Il sistema di controllo della versione utilizzata dal manutentore precedente potrebbe modifiche utili, quindi potrebbe essere una buona idea dare un'occhiata lì. Controllare se il file `controllo` del pacchetto precedente conteneva intestazioni di collegamento al sistema di controllo della versione per il pacchetto e se esiste ancora.

La rimozione di pacchetti da `unstable` (not `testing`, `stable` or `oldstable`) fa scattare la chiusura di tutti i bug relativi al pacchetto . Si dovrebbe guardare attraverso tutti i bug chiusi (compresi bug archiviati) ed estrarre e riaprire qualunque sia stato chiuso in una versione che termina in `+rm` e applicare nuovamente. Qualunque che non si applichi deve essere contrassegnato come risolto nella versione corretta se si è a conoscenza.

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 Fare port e port del proprio lavoro

Debian supporta un numero sempre crescente di architetture. Anche se non si è un autore di port, e si utilizza una sola architettura, è parte del dovere di un maintainer essere a conoscenza dei problemi di portabilità. Pertanto, anche se non si è un porter, si consiglia di leggere la maggior parte di questo capitolo.

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 Siate gentili con gli autori di port

Gli autori di port hanno un compito difficile e unico, poiché sono necessari per affrontare una grande mole di pacchetti. Idealmente, ogni pacchetto sorgente dovrebbe potersi compilare così come è. Purtroppo, spesso questo non è vero. Questa sezione contiene un elenco di «cose da tenere d'occhio» spesso committitate dai maintainer Debian: problemi comuni che spesso ostacolano gli autori di port, e rendono il loro lavoro inutilmente difficile.

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.

Di gran lunga, la maggior parte dei problemi incontrati dagli autori di port sono causati da *bug di pacchettizzazione* dei pacchetti sorgente. Ecco una lista di cose che dovreste controllare o di cui dovreste essere a conoscenza.

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*).

   Se non è possibile impostare una corretta chroot, `dpkg-depcheck` può essere di aiuto (si consulti *dpkg-depcheck*).

   Si consulti il Debian Policy Manual per le istruzioni su come impostare le dipendenze di compilazione.

2. Non si imposti l'architettura a un valore diverso da `all` o `any` a meno che non si voglia veramente farlo. In troppi casi, i maintainer non seguono le istruzioni del Debian Policy Manual. Impostare l'architettura ad una sola (come ad esempio `i386` o `amd64`) è di solito non corretto.

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. Assicuratevi di non distribuire il pacchetto sorgente con il `debian/files` o `debian/substvars`. Essi devono essere rimossi dal target `clean` di `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. Non dipendere dal fatto che il pacchetto che si sta compilando sia già installato (un sotto-caso del problema di cui sopra). Ci sono, naturalmente, eccezioni a questa regola, ma si sia consapevoli che qualsiasi caso come questo necessita di bootstrap manuale e dagli strumenti per compilazione automatica dei pacchetti.

7. Non fate affidamento su una particolare versione del compilatore, se possibile. In caso contrario, assicurarsi che le dipendenze di compilazione rispecchino le restrizioni, anche se probabilmente si andrà incontro a guai, poiché diverse architetture a volte si standardizzano su compilatori diversi.

8. Assicurarsi che il proprio `debian/rules` contenga target separati per `binary-arch` e `binary-indep`, come richiede il Debian Policy Manual. Assicurarsi che entrambi i target lavorino indipendentemente, cioè, che sia possibile chiamare il target senza aver prima chiamato l'altro. Per verificarlo, provate ad eseguire `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 Linee guida per i caricamenti degli autori di port

Se il pacchetto si compila senza modifiche per l'architettura per la quale si sta facendo il port, si è fortunati e il proprio compito è semplice. Questa sezione si applica a questo caso, descrive come compilare e caricare il pacchetto binario in modo che sia correttamente installato nell'archivio. Se c'è bisogno di creare una patch per il pacchetto in modo da farlo compilare per le altre architetture, si sta effettivamente facendo un NMU di un sorgente, dunque si consultino le *Quando e come fare un NMU*.

Per un caricamento effettuato da un autore di port, nessuna modifica deve essere stata fatta al sorgente. Non è necesssario toccare alcun file nel pacchetto sorgente. Questo include `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 Ricompilazione o NMU dei soli binari

A volte il primo caricamento di un autore di port è problematico perché l'ambiente in cui il pacchetto è stato compilato non era abbastanza buono (librerie datate o obsolete, un compilatore non buono, etc.). Allora si può solo aver necessità di ricompilarlo in un ambiente aggiornato. Tuttavia, è necessario incrementare il numero della versione, in questo caso, in modo che il vecchio non buono pacchetto possa essere sostituito nell'archivio Debian (`dak` si rifiuta di installare nuovi pacchetti, se non hanno un numero di versione maggiore di quella attualmente disponibile).

Si deve fare in modo che il proprio NMU di soli binari non renda il pacchetto non installabile. Questo potrebbe accadere quando un pacchetto sorgente genera pacchetti dipendenti e non dall'architettura che possiedono interdipendenze generate utilizzando la variabile di sostituzione di dpkg `$(Source-Version)`.

Nonostante la modifica necessaria del changelog, questi sono chiamati NMU di soli binari: non è necessario in questo caso far sì che tutte le altre architetture si considerino non aggiornate o con necessità di ricompilazione.

Tali ricompilazioni richiedono un particolare numero di versione «magico», in modo che gli strumenti di manutenzione dell'archivio riconoscano che, anche se vi è una nuova versione di Debian, non vi è alcun aggiornamento di sorgenti corrispondente. Se si sbaglia, i manutentori dell'archivio respingeranno il proprio caricamento (per mancanza del corrispondente codice sorgente).

Il «magico» per un NMU con sola ricompilazione viene attivato utilizzando un suffisso aggiunto al numero di versione del pacchetto, seguendo la forma b*numero*. Per esempio, se l'ultima versione sulla quale si sta ricompilando era la versione `2.9-3`, l'NMU solo binario dovrebbe avere la versione `2.9-3+b1`. Se l'ultima versione era `3.4+b1` (cioè un pacchetto nativo con un precedente NMU con ricompilazione), il proprio NMU solo binario dovrebbe avere un numero di versione `3.4+b2`.[2]

In modo simile ai caricamenti iniziali dell'autore del port, il modo corretto di invocare `dpkg-buildpackage` è `dpkg-buildpackage -B` per compilare solo le parti del pacchetto dipendenti dell'architettura.

5.12.2.2 Quando fare un NMU del sorgente se si è un autore di port

Gli autori di port generalmente nel fare un NMU del sorgente seguono le linee guida presenti in *Caricamenti dei Non-Maintainer (NMU)*, proprio come i non-porter. Tuttavia, ci si aspetta che il ciclo di attesa per un NMU del sorgente di un autore di port sia minore di quello per chi non è autore di port, poiché i gli autori di port devono gestire una grande quantità di pacchetti. Di nuovo, la situazione varia a seconda della distribuzione sulla quale si stanno effettuando i caricamenti. Essa varia anche se l'architettura è candidata ad essere inclusa nel prossimo rilascio stabile; i gestori dei rilasci decidono e annunciano quali architetture sono candidate.

Se si è un autore di port che fa un NMU per `unstable`, le linee guida di cui sopra per la portabilità dovrebbero essere seguite, con due varianti. In primo luogo, il periodo di attesa accettabile, il tempo tra quando il bug è presentato al BTS e quando è considerato valido per fare un NMU, è di sette giorni per gli autori di port che lavorano sulla distribuzione `unstable`. Questo periodo può essere abbreviato se il problema è critico e crea difficoltà sul lavoro di creazione dei port, a discrezione del gruppo di lavoro sui port. (Ricordate, niente di tutto ciò è Policy, ma solo linee guida stabilite di comune accordo.) Per caricamenti su `stable` o su `testing`, coordinarsi prima con l'appropriato team di rilascio.

In secondo luogo, gli autori di port che fanno NMU di sorgenti dovrebbero assicurarsi che il bug presentato al BTS dovrebbe essere di gravità `serious` o superiore. Questo assicura che un singolo pacchetto sorgente possa essere usato per compilare ogni architettura supportata da Debian al momento del rilascio. È molto importante avere una versione del pacchetto binario e di quello del sorgente per tutte le architetture al fine di conformarsi con molte licenze.

Gli autori dei port dovrebbero cercare di evitare le patch che semplicemente aggirano i bug nella versione attuale dell'ambiente di compilazione, kernel, oppure libc. A volte tali stratagemmi non possono essere evitati. Se è necessario aggirare bug del compilatore e simili, assicurarsi di `#ifdef` il proprio lavoro correttamente; inoltre, documentare le modifiche fatte per aggirare i bug in modo che la gente sappia di rimuoverli una volta che i problemi esterni siano stati corretti.

Gli autori di port possono anche avere un luogo non ufficiale dove possono mettere i risultati del loro lavoro durante il periodo di attesa. Questo aiuta gli altri che utilizzano il port ad avere il vantaggio del lavoro del suo autore, anche durante il periodo di attesa. Naturalmente, tali luoghi non hanno alcuna benedizione o status ufficiale, quindi stare attenti.

5.12.3 Infrastrutture e automazione per il port

C'è un'infrastruttura e diversi strumenti per aiutare ad automatizzare il port dei pacchetti. Questa sezione contiene una breve panoramica di questi strumenti di automazione e di port; si consulti la documentazione dei pacchetti o i riferimenti per informazioni complete.

5.12.3.1 Mailing list e pagine web

Le pagine Web che contengono lo stato di ogni porto sono disponibili all'indirizzo https://www.debian.org/ports/.

Ogni port di Debian ha una mailing list. L'elenco delle mailing list dedicate ai port può essere trovato su https://lists.debian.org/ports.html. Questi elenchi vengono utilizzati per coordinare gli autori di port, e per collegare gli utenti di un dato port con gli autori.

[2] In passato, tali NMU utilizzarono il numero di terzo livello della parte della revisione di Debian per indicare solo la loro ricompilazione; tuttavia, questa sintassi era ambigua con i pacchetti nativi e non ha permesso una corretta ordinazione delle sole ricompilazioni NMU, NMU sorgente, e NMU di sicurezza sullo stesso package, ed è stata abbandonata in favore di questa nuova sintassi.

5.12.3.2 Strumenti per gli autori di port

Le descrizioni di diversi strumenti per il port si possono trovare su *Strumenti per i port*.

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.

Siamo molto orgogliosi di questo sistema, dal momento che ha tanti usi possibili. Gruppi di sviluppo indipendenti possono utilizzare il sistema per diverse sotto-versioni di Debian, che possono o non possono essere di interesse generale (per esempio, una versione di Debian compilata con il bound-checking di gcc). Essa consentirà inoltre a Debian di ricompilare intere distribuzioni rapidamente.

Il team di wanna-build, responsabile dei buildd, può essere raggiunto all'indirizzo debian-wb-team@lists.debian.org. Per determinare chi contattare (team di wanna-build, team del rilascio) e come (posta, BTS), si faccia riferimento a https://lists.debian.org/debian-project/2009/03/msg00096.html.

Quando si richiedono dei binNMUs o dei give-backs (un nuovo tentativo dopo una compilazione fallita), utilizzare il formato descritto in https://release.debian.org/wanna-build.txt.

5.12.4 Quando il pacchetto *non* è portabile

Alcuni pacchetti hanno ancora problemi con la compilazione o con il funzionamento su alcune delle architetture supportate da Debian, ed è del tutto impossibile farne il port, o non entro un ragionevole lasso di tempo. Un esempio è un pacchetto che è SVGA-specifico (disponibile solo per i386 e amd64), o che utilizzi altre caratteristiche specifiche dell'hardware non supportate su tutte le architetture.

Al fine di evitare che pacchetti danneggiati vengano caricati nell'archivio, e sprecare tempo dei buildd, è necessario fare un paio di cose:

- In primo luogo, assicurarsi che il pacchetto *fallisca* la compilazione su architetture che non supporta. Ci sono alcuni modi per raggiungere questo obiettivo. Il modo migliore è quello di avere una piccola suite di test che ne verifichi le funzionalità durante la compilazione, e che fallirà se non funziona. Questa è comunque una buona idea, in modo da evitare (alcuni) caricamenti difettosi in tutte le architetture, e permetterà anche al pacchetto di compilarsi appena la funzionalità richiesta viene resa disponibile.

  Inoltre, se si ritiene che l'elenco delle architetture supportate sia piuttosto costante, si dovrebbe cambiare any in un elenco delle architetture supportate in debian/control. In questo modo, anche la compilazione fallirà, e lo indicherà ad un lettore umano senza che realmente la si tenti.

- Al fine di evitare che gli strumenti di compilazione automatica cerchino dal cercare inutilmente di compilare il pacchetto, deve essere incluso in Packages-arch-specific, un elenco utilizzato dallo script wanna-build. L'attuale versione è disponibile come https://wiki.debian.org/PackagesArchSpecific; consultare la parte iniziale del file per chi contattare per le modifiche.

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 Marcare i pacchetti non-free come compilabili automaticamente

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. Si controlli se è legalmente consentito e tecnicamente possibile automatizzare la compilazione del pacchetto;
2. Si aggiunga `XS-Autobuild: yes` nell'intestazione di `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 Caricamenti dei Non-Maintainer (NMU)

Ogni pacchetto ha uno o più maintainer. Normalmente, queste sono le persone che ci lavorano e caricano nuove versioni del pacchetto. In alcune situazioni è utile che anche altri sviluppatori possano caricare una nuova versione, per esempio se vogliono risolvere un bug in un pacchetto che non mantengono, quando il maintainer ha bisogno di aiuto per rispondere alle segnalazioni. Tali aggiornamenti sono chiamati *Non-Maintainer Upload (NMU)*.

5.13.1 Quando e come fare un NMU

Prima di fare un NMU, si considerino le seguenti domande:

- 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.
- Quanto si è sicuri delle modifiche? Si ricordi il giuramento di Ippocrate: «Prima di tutto, non nuocere.» È meglio lasciare un pacchetto con un bug grave aperto che applicare una patch non funzionante, o una che nasconde il bug invece di risolverlo. Se non si è sicuri al 100% di quello che si è fatto, potrebbe essere una buona idea chiedere il parere di altri. Si ricordi che se si rompe qualcosa nel proprio NMU, molte persone saranno molto scontente al riguardo.
- Come si e sicuri sulle modifiche? Si ricorda il giuramento di Ippocrate: "Soprattutto, non nuocere". E' meglio lasciare un pacchetto con un grave bug aperto che applicare una patch non funzionante, o uno che nasconde il bug invece di risolverlo. Se non si è sicuri al 100% di quello che si è fatto, potrebbe essere una buona idea chiedere il parere di altri. Ricordate che se si rompe qualcosa nel proprio NMU, molte persone saranno molto infelici di questo.
- Have you clearly expressed your intention to NMU, at least in the BTS? If that didn't generate any feedback, it might also be a good idea to try to contact the maintainer by other means (email to the maintainer addresses or private email, IRC).
- Se il maintainer è in genere attivo e reattivo, si è provato a contattarlo? Generalmente, dovrebbe essere considerato preferibile che i maintainer si prendano cura di un problema essi stessi e che abbiano la possibilità di rivedere e correggere la patch, in quanto sono più consapevoli dei potenziali problemi che un autore di NMU potrebbe non considerare. Spesso è un miglior uso del tempo di tutti se al maintainer viene data la possibilità di caricare una propria soluzione.

Nel fare un NMU, è necessario assicurarsi che la propria intenzione di un NMU sia chiara. Quindi, è necessario inviare una patch al BTS con le differenze tra il pacchetto attuale e il proprio NMU. Lo script `nmudiff` nel pacchetto `devscripts` potrebbe essere utile.

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`.

A meno che non si disponga di un ottimo motivo per non farlo, è necessario quindi dare un po' di tempo al maintainer per reagire (per esempio, caricandolo nella coda `DELAYED`). Ecco alcuni valori consigliati da usare nei casi differiti:

- Caricamento che risolve solo bug critici per il rilascio più vecchi di 7 giorni, senza alcuna attività del maintainer sul bug per 7 giorni e nessuna indicazione che si stia lavorando ad una soluzione: 0 giorni
- Caricamento che risolve solo bug critici per il rilascio più vecchi di 7 giorni: 2 giorni
- Caricamento che risolve solo bug critici per il rilascio e per bug importanti: 5 giorni
- Other NMUs: 15 days

Tali ritardi sono solo degli esempi. In alcuni casi, come ad esempio caricamenti di correzioni di problemi di sicurezza o di correzioni di bug banali che bloccano una transizione, è auspicabile che il pacchetto corretto raggiunga prima `unstable`.

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).

Questa non è una licenza per eseguire NMU in modo sconsiderato. Se si effettua un NMU quando è chiaro che i maintainer sono attivi e avrebbero preso in considerazione una patch tempestivamente, oppure se si ignorano le raccomandazioni di questo documento, il caricamento potrebbe essere causa di conflitto con il maintainer. Si dovrebbe sempre essere pronti a difendere la saggezza di ogni NMU che si fa sulla base dei suoi meriti.

5.13.2 NMU e `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.
```

Il modo di assegnare versioni agli NMU è differente per i pacchetti nativi e per i non-nativi.

Se il pacchetto è un pacchetto nativo (senza una revisione Debian nel numero di versione), la versione deve essere la versione dell'ultimo caricamento del maintainer, più `+nmu`*X*, dove *X* è un contatore a partire 1. Se anche l'ultimo caricamento è stato un NMU, il contatore dovrebbe essere aumentato. Ad esempio, se la versione corrente è `1.5`, allora un NMU avrebbe la versione `1.5+nmu1`.

Se il pacchetto non è un pacchetto nativo, si dovrebbe aggiungere un numero di versione secondario alla parte di revisione Debian del numero di versione (la parte dopo l'ultimo trattino). Questo numero aggiuntivo deve iniziare da 1. Ad esempio, se la versione corrente è `1.5-2`, poi un NMU otterrebbe la versione `1.5-2.1`. Se nell'NMU viene pacchettizzata una nuova versione a monte, la revisione Debian viene impostata a `0`, per esempio `1.6-0.1`.

In entrambi i casi, se anche l'ultimo caricamento è stato un NMU, il contatore dovrebbe essere aumentato. Ad esempio, se la versione corrente è `1.5+nmu3` (un pacchetto nativo che ha già subito un NMU), l'NMU avrebbe versione `1.5+nmu4`.

Uno speciale schema per i nomi di versione del codice è necessario per evitare di danneggiare il lavoro del maintainer, in quanto utilizzare un numero intero per la revisione Debian potenzialmente potrebbe entrare in conflitto con un caricamento del maintainer già in preparazione al momento di un NMU, o anche con uno presente nella coda NEW dell'ftp. Ha anche il vantaggio di rendere visivamente chiaro che un pacchetto nell'archivio non è stato fatto dal maintainer ufficiale.

Se si carica un pacchetto su testing o stable, a volte è necessario fare il «fork» dell'albero dei numeri di versione. Questo è il caso dei caricamenti di sicurezza, ad esempio. Per questo dovrebbe essere usata una versione della forma `+deb`*XY*`u`*Z*, dove *X* e *Y* sono i numeri di versione principale e minore, e *Z* è un contatore che parte da 1.

Quando il numero di rilascio non è ancora noto (spesso è il caso per `testing`, all'inizio dei cicli di rilascio), deve essere utilizzato il più basso numero di versione che è più alto dell'ultimo numero di rilascio stabile. Per esempio, mentre Lenny (Debian 5.0) è stabile, un NMU di sicurezza a stable per un pacchetto alla versione `1.5-3` avrebbe versione `1.5-3+deb50u1`, mentre un NMU di sicurezza per Squeeze avrebbe versione `1.5-3+deb60u1`. Dopo l'uscita di Squeeze, i caricamenti di sicurezza per la distribuzione `testing` saranno avranno versione `+deb61uZ`, fino a quando non è noto se tale versione sarà Debian 6.1 o Debian 7.0 (se si avvera quest'ultimo caso i caricamenti avranno versione come `+deb70uZ`).

5.13.3 Utilizzare la coda `DELAYED/`

Il dover attendere una risposta dopo che si è richiesto il permesso per un NMU è inefficiente, perché costa all'autore dell'NMU un cambio di contesto per ritornare sul problema. La coda `DELAYED` (si consulti *Caricamenti differiti*) consente allo sviluppatore che fa l'NMU di svolgere al tempo stesso tutti i compiti necessari. Per esempio, invece di dire al maintainer che si caricherà il pacchetto aggiornato tra 7 giorni, si dovrebbe caricare il pacchetto in `DELAYED/7` e dire al maintainer che ha 7 giorni di tempo per reagire. Durante questo tempo, il maintainer può chiedere di ritardare il caricamento di un po', o annullarlo.

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.

La coda `DELAYED` non deve essere utilizzata per mettere ulteriore pressione sul maintainer. In particolare, è importante che ci si renda disponibili ad annullare o ritardare il caricamento prima della scadenza del ritardo in quanto il maintainer non lo può annullare da solo.

Se si fa un NMU `DELAYED` e il maintainer aggiorna il pacchetto prima della scadenza del ritardo, il caricamento sarà respinto perché una nuova versione è già disponibile nell'archivio. Idealmente, il maintainer avrà cura di includere in questa versione le modifiche proposte (o almeno una soluzione per i problemi che affrontano).

5.13.4 NMU dal punto di vista del maintainer

Quando qualcuno fa un NMU per il vostro pacchetto, questo significa che vuole aiutare a mantenerlo in buona forma. Questo dà agli utenti più velocemente dei pacchetti corretti. Si può considerare di chiedere all'autore dell'NMU di diventare un co-maintainer del pacchetto. La ricezione di un NMU su un pacchetto non è una cosa negativa; ma semplicemente significa che il pacchetto è abbastanza interessante per le altre persone da spingerle a lavorare su di esso.

Per riconoscere un NMU, si includano i suoi cambiamenti e le voci del changelog nel proprio prossimo caricamento da maintainer. Se non si riconosce l'NMU non includendo le voci del changelog dell'NMU nel proprio changelog, i bug rimarranno chiusi nel BTS, ma saranno elencati come presenti nella propria versione di maintainer del pacchetto.

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 Source NMUs vs Binary-only NMUs (binNMUs)

Il nome completo di un NMU è *NMU sorgente*. C'è anche un altro tipo, vale a dire la *NMU solo binario*, o *binNMU*. Un binNMU è anche un pacchetto caricato da una persona diversa dal maintainer del pacchetto. Tuttavia, è un solo un caricamento dei soli binari.

Quando una libreria (o altra dipendenza) viene aggiornata, i pacchetti che la utilizzano possono aver bisogno di essere ricompilati. Dal momento che non sono necessarie modifiche al sorgente, viene utilizzato lo stesso pacchetto sorgente.

I binNMU sono di solito attivati sui buildd da wanna-build. Viene aggiunta una voce al `debian/changelog`, che spiega perché il caricamento è stato necessario e incrementa il numero di versione come descritto in *Ricompilazione o NMU dei soli binari*. Questa voce non deve essere inclusa nel prossimo caricamento.

Buildds carica i pacchetti per le loroarchitetture negliarchivi come upload binari. Parlando più precisamente, questi sono binNMUs. Comunque, non sono chimamati NMU, e non sono aggiunte voci al `debian/changelog`.

5.13.6 NMU e caricamenti di QA

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.

I caricamenti di QA sono molto simili a normali caricamenti del maintainer: possono correggere qualsiasi cosa, anche problemi minori; la numerazione delle versioni è normale, e non vi è alcuna necessità di utilizzare un caricamento differito. La differenza è che non si viene elencati come il `Maintainer` o `Uploader` per il pacchetto. Inoltre, la voce del changelog del caricamento di QA ha una speciale prima riga:

```
* QA upload.
```

Se si vuole fare un NMU, e sembra che il maintainer non sia attivo, è consigliabile controllare se il pacchetto sia orfano (questa informazione viene visualizzata sulla pagina Package Tracking System del pacchetto). Quando si fa il primo caricamento di QA ad un pacchetto orfano, il maintainer deve essere impostato su `Debian QA Group <packages@qa.debian.org>`. I pacchetti orfani che non avevano ancora un caricamento QA hanno ancora indicato il loro vecchio maintainer. C'è un elenco di questi ultimi in 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 *L'adozione di un pacchetto*).

5.13.7 NMU e caricamenti del team

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 *La manutenzione collaborativa*) 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 *Caricamenti dei Non-Maintainer (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 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.

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

5.15 La manutenzione collaborativa

Manutenzione collaborativa è un termine che descrive la condivisione dei compiti di manutenzione dei pacchetti Debian tra più persone. Questa collaborazione è quasi sempre una buona idea, dato che in genere si traduce in una maggiore qualità e in tempi più rapidi per la soluzione di bug. Si raccomanda vivamente che i pacchetti con priorità `standard` o che sono parte dell'insieme base abbiano dei co-maintainer.

In generale vi è un maintainer primario e uno o più co-maintainer. Il maintainer primario è la persona il cui nome è inserito nel campo `Maintainer` del file `debian/control`. Co-maintainer sono tutti gli altri maintainer, di solito elencati nel campo `Uploaders` del file `debian/control`.

Nella sua forma più semplice, il processo di aggiunta di un nuovo co-maintainer è abbastanza facile:

- 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.
- Si aggiunga il nome corretto del co-maintainer e l'indirizzo nel campo `Uploaders` nel primo paragrafo del file `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*).

Un'altra forma di manutenzione collaborativa è la manutenzione in un team, che è raccomandata se si mantengono diversi pacchetti con lo stesso gruppo di sviluppatori. In tal caso, i campi `Maintainer` e `Uploaders` di ogni pacchetto devono essere gestiti con attenzione. Si consiglia di scegliere tra i seguenti due schemi:

1. Inserire il membro del team che è il principale responsabile per il pacchetto nel campo `Maintainer`. Nel campo `Uploaders`, inserire l'indirizzo della mailing list, ed i membri del team che si interessano al pacchetto.
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).

In ogni caso, è una cattiva idea mettere automaticamente tutti i membri del team nel campo `Uploaders`. Ciò ingombra l'elenco dei Developer's Package Overview (si consulti *Panoramica dei pacchetti per sviluppatori*) con pacchetti di cui di fatto uno non si occupa veramente, e crea un falso senso di buona manutenzione. Per lo stesso motivo, i membri del team non devono di aggiungere se stessi nel campo `Uploaders` solo perché stanno caricando il pacchetto una sola volta, possono fare un «caricamento di Team» (si consulti *NMU e caricamenti del team*). Al contrario, è una cattiva idea tenere un pacchetto con il solo indirizzo della mailing list come `Maintainer` e senza nessuno `Uploaders`.

5.16 La distribuzione testing

5.16.1 Nozioni di base

I pacchetti sono generalmente installati nella distribuzione `testing` dopo aver subito un periodo di prova in `unstable`.

Essi devono essere sincronizzati su tutte le architetture e non devono avere dipendenze tali da renderli non installabili; inoltre devono generalmente non avere bug critici per il rilascio noti nel momento dell'installazione in `testing`. In questo modo, `testing` dovrebbe essere sempre vicina ad essere candidata al rilascio. Si veda sotto per i dettagli.

5.16.2 Aggiornamenti da unstable

Gli script che aggiornano la distribuzione `testing` vengono eseguiti due volte al giorno, subito dopo l'installazione dei pacchetti aggiornati; questi script si chiamano `britney`. Essi generano i file `Packages` per la distribuzione `testing`, ma lo fanno in modo intelligente; cercano di evitare qualsiasi incoerenza e di utilizzare solo i pacchetti senza bug.

L'inclusione di un pacchetto da `unstable` è subordinata alle seguenti:

- 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;
- Non deve avere nuovi bug critici per il rilascio (bug RC che riguardano la versione disponibile in `unstable`, ma non intaccano la versione in `testing`);
- Deve essere disponibile su tutte le architetture sulle quali è stato precedentemente compilato in `unstable`. *L'utility dak ls* può essere utile per verificare queste informazioni;
- Non deve rendere diffettosa alcuna dipendenza di un pacchetto che è già disponibile in `testing`;
- I pacchetti da cui dipende devono essere disponibili in `testing` o devono essere accettati in `testing` nello stesso momento (e lo saranno se soddisfano tutti i criteri necessari);
- La fase del progetto. Cioè transizioni automatiche sono disattivate durante il *freeze* della distribuzione `testing`.

Per sapere se un pacchetto è in fase di passaggio a `testing` o meno, si consulti il prodotto dello script di `testing` nella pagina web della distribuzione testing, oppure si utilizzi il programma `grep-excuses`, che è nel pacchetto `devscripts`. Questa utility può essere facilmente utilizzata in un crontab5 per tenersi informati sull'avanzamento dei propri pacchetti in `testing`.

Il file `update_excuses` non sempre dà il motivo preciso per cui il pacchetto è stato rifiutato; potrebbe essere necessario trovarlo da soli, cercando ciò che sarebbe stato reso difettoso dall'inclusione. La pagina web di testing dà qualche informazione in più sulle problematiche comuni che possono causare questi problemi.

A volte, alcuni pacchetti non entrano mai in `testing`, perché l'insieme di interrelazioni è troppo complicato e non può essere risolto dagli script. Si veda sotto per i dettagli.

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 Obsoleti

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`.

Si consideri questo esempio:

	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`.

Tuttavia, se l'ftp-master rimuove un pacchetto in `unstable` (qui in `arm`):

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

In questo caso, il pacchetto è aggiornato su tutte le architetture di rilascio in `unstable` (e quella aggiuntiva `hurd-i386` non importa, considerato che non è un'architettura inclusa nel rilascio).

A volte, la questione che viene sollevata è se è possibile far avanzare pacchetti che non sono ancora stati compialti su tutte le architetture: No. Sempre e comunque no. (Tranne se si mantiene glibc o giù di lì.)

5.16.2.2 Rimozioni da testing

A volte, un pacchetto viene rimosso per consentire ad un altro pacchetto di entrare: questo accade solo per permettere ad *un altro* pacchetto di avanzare se è pronto sotto ogni altro aspetto. Supponiamo ad esempio che `a` non può essere installato con la nuova versione di `b`; allora `a` può essere rimosso per consentire a `b` di entrare.

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).

Inoltre, se un pacchetto è stato rimosso da `unstable`, e nessun pacchetto in `testing` dipende più da esso, allora sarà automaticamente rimosso.

5.16.2.3 Dipendenze circolari

Una situazione che non è gestita molto bene da britney è se un pacchetto `a` dipende dalla nuova versione del pacchetto `b`, e viceversa.

Un esempio di questo è:

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

Né il pacchetto `a`, né il pacchetto `b` sono considerati per l'aggiornamento.

Attualmente, questo richiede qualche suggerimento manuale al team di rilascio. contattarli con l'invio di una email a `debian-release@lists.debian.org` se ciò dovesse accadere ad uno dei propri pacchetti.

5.16.2.4 Influenza del pacchetto in testing

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.

In sintesi questo significa: l'unica influenza che un pacchetto presente in `testing` ha su una nuova versione dello stesso pacchetto è che la nuova versione potrebbe entrarci più facilmente.

5.16.2.5 Dettagli

Se si è interessati a maggiori dettagli, cosi è come funziona britney:

I pacchetti sono osservati per determinare se essi sono validi candidati. Da questo derivano le motivazioni per gli aggiornamenti. Le ragioni più comuni per cui un pacchetto non viene considerato essere troppo giovane, mancanza di bug RC, e obsoleto su alcune architetture. Per questa parte di britney, i gestori dei rilasci posseggono martelli di varie dimensioni, chiamati suggerimenti (si consulti sotto), per forzare britney a considerare un pacchetto.

Ora arriva la parte più complessa: britney tenta di aggiornare `testing` con i candidati validi. Per questo, britney cerca di aggiungere ogni candidato valido per la distribuzione testing. Se il numero di pacchetti non installabili in `testing` non aumenta, il pacchetto viene accettato. Da quel momento in poi, il pacchetto viene considerato parte di `testing`, in modo che tutti i test di installabilità successivi includeranno questo pacchetto. Suggerimenti da parte del team di rilascio vengono elaborati prima o dopo questo ciclo principale, a seconda del tipo esatto.

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 *Aggiornamenti diretti di testing* or override the urgency.

5.16.3 Aggiornamenti diretti di testing

La distribuzione `testing` è alimentata con i pacchetti da `unstable` in base alle regole sopra esposte. Tuttavia, in alcuni casi, è necessario caricare i pacchetti compilati solo per `testing`. Per questo, è meglio fare il caricamento in `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.

I numeri della versione sono solitamente individuati concatenando +deb*X*u*Y*, dove *X* è il numero maggiore della release Debian e *Y* è un contatore che inizia da 1. e.s.`1:2.4.3-4+deb13u1`.

Assicurarsi di non essersi dimenticata nessuna di queste cose nel proprio caricamento:

- Assicurarsi che il proprio pacchetto abbia davvero bisogno di passare attraverso `testing-proposed-updates`, e non possa passare attraverso `unstable`;
- Assicurarsi di includere solo la quantità minima di modifiche;
- Assicurarsi di aver incluso una spiegazione adeguata nel changelog;
- Make sure that you've written the testing *Nomi in codice dei rilasci* (e.g. `forky`) into your target distribution;
- Assicurarsi di aver compilato e testato il proprio pacchetto in `testing`, non in `unstable`;
- Assicurarsi che il numero di versione sia più alto rispetto alla versione in `testing` e in `testing-proposed-updates`, e inferiore a quello in `unstable`;
- Ask for authorization for uploading from the release managers.
- Dopo aver effettuato il caricamento e dopo che la compilazione è riuscita su tutte le piattaforme, si contatti il team di rilascio su `debian-release@lists.debian.org` chiedendogli di approvare il caricamento.

5.16.4 Domande frequenti

5.16.4.1 Quali sono i bug critici per il rilascio e come vengono contati?

Tutti i bug di alcuni livelli più elevati di gravità sono di base considerati release-critical; attualmente, questi sono i bug `critical`, `grave` e `serious`.

Tali bug sono ritenuti avere un impatto sulle possibilità che il pacchetto venga rilasciato con la distribuzione `stable` di Debian: in generale, se un pacchetto ha un bug critico per il rilascio aperto, non sarà integrato in `testing`, e di conseguenza non sarà rilasciato in `stable`.

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 Come potrebbe l'installazione di un pacchetto in `testing` rendere difettosi altri pacchetti?

La struttura degli archivi di distribuzione è tale che possono contenere solo una versione di un pacchetto; un pacchetto è definito dal suo nome. Così, quando il pacchetto sorgente `acmefoo` è installato in `testing`, insieme con i suoi pacchetti binari `acme-foo-bin`, `acme-bar-bin`, `libacme-foo1` e `libacme-foo-dev`, la versione precedente viene rimossa.

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.

Se si hanno problemi con gruppi complessi di pacchetti come questo, si contatti `debian-devel@lists.debian.org` o `debian-release@lists.debian.org` per un aiuto.

5.17 The Stable backports archive

5.17.1 Nozioni di base

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).

CAPITOLO 6

Buone pratiche per la pacchettizzazione

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.

Questo capitolo fornisce alcune buone pratiche per gli sviluppatori Debian. Tutte le raccomandazioni sono solo tali, e non sono requisiti o policy. Questi sono solo alcuni spunti soggettivi, i consigli e i punti raccolti da sviluppatori Debian. Ci si senta liberi di scegliere quello che funziona meglio.

6.1 Buone pratiche per `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 Script di supporto

L'idea nell'utilizzo degli script di aiuto in `debian/rules` è che essi hanno consentito ai maintainer di usare e condividere la logica comune tra molti pacchetti. Si prenda per esempio la questione dell'installazione delle voci di menu: è necessario mettere il file in `/usr/share/menu` (o `/usr/lib/menu` per gli eseguibili binari dei menufile, se questo è necessario), e aggiungere i comandi agli script del maintainer per registrare ed annullare la registrazione delle voci di menu. Dal momento che questa è una cosa molto comune da fare con i pacchetti, perché ogni maintainer dovrebbe riscrivere tutto questo da solo, a volte con bug? Inoltre, supponendo che la cartella menu cambi, ogni pacchetto dovrebbe essere cambiato.

Gli script di supporto si occupano di questi problemi. Supponendo che ci si attenga alle convenzioni previste dallo script di supporto, quest'ultimo si prende cura di tutti i dettagli. Cambiamenti nella policy possono essere effettuati nello script helper; successivamente i pacchetti avranno solo bisogno di essere ricompilati con la nuova versione dell'helper e nessuna ulteriore modifica.

Panoramica degli strumenti del Debian Maintainer contiene un paio di diversi script di supporto. Il sistema di supporto più comune e migliore (a nostro parere) è `debhelper`. I sistemi di supporto precedenti, come `debmake`, erano monolitici: non si poteva scegliere quale parte dell'helper si riteneva utile, ma si doveva usare l'helper per fare tutto. `debhelper`, invece, è una serie di programmi piccoli e separati `dh_*`. Per esempio, `dh_installman` installa e comprime le pagine man, `dh_installmenu` installa i file di menu, e così via. Così, si offre sufficiente flessibilità per essere in grado di utilizzare i piccoli script di aiuto, dove utile, in abbinamento con i comandi manuali in `debian/rules`.

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 Pacchetti binari multipli

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).

Il secondo caso può essere facilmente gestito in `debian/rules`. Bisogna solo spostare i file appropriati dalla cartella di compilazione in alberi temporanei del pacchetto. È possibile farlo utilizzando `install` o `dh_install` da `debhelper`. Ci si assicuri di controllare le diverse permutazioni dei vari pacchetti, assicurandosi di avere il corretto insieme di dipendenze tra pacchetti in `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 Buone pratiche per `debian/control`

Le seguenti pratiche sono rilevanti per il file`debian/control`. Esse integrano la Policy sulla descrizione dei pacchetti.

La descrizione del pacchetto, come definito dal corrispondente campo nel file `control`, contiene sia la sinossi del pacchetto sia la descrizione lunga del pacchetto. *Linee guida generali per le descrizioni dei pacchetti* descrive le linee guida comuni ad entrambe le parti della descrizione del pacchetto. In seguito, *La sinossi del pacchetto, o una breve descrizione* fornisce specifiche linee guida per la sinossi e *La descrizione lunga* contiene specifiche linee guida per la descrizione.

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 Linee guida generali per le descrizioni dei pacchetti

La descrizione del pacchetto dovrebbe essere scritta probabilmente per l'utente medio, la persona media che utilizzerà ed avrà benefici dal pacchetto. Per esempio, pacchetti di sviluppo sono per gli sviluppatori e possono essere di natura tecnica nella loro linguaggio. Applicazioni più generiche, come gli editor, dovrebbero essere scritte per un utente meno tecnico.

La nostra analisi delle descrizioni dei pacchetti ci porta a concludere che la maggior parte delle descrizioni dei pacchetti sono di natura tecnica, cosi è, non sono scritte per avere senso per gli utenti non tecnici. A meno che il proprio pacchetto non sia realmente solo per utenti tecnici, questo costituisce un problema.

Come si scrive per gli utenti non tecnici? Si eviti il gergo. Evitare di riferirsi ad altre applicazioni o framework che l'utente potrebbe non conoscere - GNOME o KDE va bene, dal momento che gli utenti hanno probabilmente

familiarità con questi termini, ma GTK probabilmente non lo è. Cercare di ipotizzare nessuna conoscenza a priori. Se è necessario utilizzare termini tecnici, spiegarli.

Si sia obiettivi. Le descrizioni dei pacchetti non sono il posto per promuovere il proprio pacchetto, non importa quanto lo si ami. Ricordare che il lettore potrebbe non essere interessato alle stesse cose che vi interessano.

I riferimenti ai nomi di eventuali altri pacchetti software, nomi di protocollo, standard o specifiche dovrebbero utilizzare la forma canonica, se ne esiste una. Ad esempio, utilizzare X Window System, X11 o X, non X Windows, X-Windows o X Window. Utilizzare GTK, non GTK+ o gtk. Usare GNOME, non Gnome. Utilizzare PostScript, non Postscript o postscript.

Se si hanno problemi di scrittura della descrizione, si potrebbe desiderare di inviarla all'indirizzo di posta elettronica `debian-l10n-english@lists.debian.org` richiedendo un parere.

6.2.3 La sinossi del pacchetto, o una breve descrizione

La policy dice che la linea di sinossi (la breve descrizione) deve essere concisa ma anche informativa, non ripetendo il nome del pacchetto.

Le sinossi funziona come una frase che descrive il pacchetto, non una frase completa, perciò la punteggiatura è inappropriata: non ha bisogno di lettere maiuscole in più o un punto finale (punto). Dovrebbe anche omettere qualsiasi iniziale articolo indefinito o definito - «a», «un» o «il». Così, per esempio:

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

Tecnicamente si tratta di un sintagma nominale senza articoli, al contrario di una frase verbale. Una buona euristica è che dovrebbe essere possibile sostituire il *nome* e la *sinossi* del pacchetto in questa formula:

Il pacchetto *name* fornisce {a, un, il, alcuni} *sinossi*.

Insiemi di pacchetti correlati possono utilizzare uno schema alternativo che divide la sinossi in due parti, la prima una descrizione di tutta la suite e la seconda una sintesi del ruolo del pacchetto all'interno di essa:

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

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

Queste sinossi seguono una formula modificata. Quando un pacchetto «*name*» ha una «*suite* di sinossi (*role*) » o «*suite* - *role*», gli elementi devono essere formulati in modo che si inseriscano nella formula:

Il *name* del pacchetto fornisce {a, un, il} *role* per la *suite*.

6.2.4 La descrizione lunga

La descrizione lunga è la principale informazione sul pacchetto disponibile agli utenti prima che lo si installi. Essa dovrebbe fornire tutte le informazioni necessarie per permettere all'utente di decidere se installare il pacchetto. Si supponga che l'utente abbia già letto la sinossi del pacchetto.

La descrizione lunga deve essere composta da frasi complete ed esaustive.

Il primo paragrafo della descrizione lunga deve rispondere alle seguenti domande: che cosa fa il pacchetto? in che modo aiuta l'utente ad assolvere ai suoi task? È importante descrivere ciò in maniera non tecnica, salvo ovviamente quando il pacchetto è destinato ad una utenza tecnica.

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)?

Fare attenzione ad evitare errori di ortografia e di grammatica. Ci si assicuri di effettuare il controllo ortografico. Entrambi `ispell` e `aspell` hanno modalità speciali per il controllo del file `debian/control`:

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

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

Gli utenti di solito si aspettano che queste domande ricevano risposta nella descrizione del pacchetto:

- Che cosa fa il pacchetto? Se si tratta di un componente aggiuntivo per un altro pacchetto, allora la breve descrizione del pacchetto per il quale è un componente aggiuntivo dovrebbe essere inserita qui.
- Perché dovrei volere questo pacchetto? Questo è legato al precedente, ma non è lo stesso (questo è un client di posta elettronica; questo è eccezionale, veloce, si interfaccia con PGP e LDAP e IMAP, ha caratteristiche X, Y e Z).
- Se il pacchetto non deve essere installato direttamente, ma è richiamato da un altro pacchetto, questo dovrebbe essere menzionato.
- Se il pacchetto è `experimental`, o ci sono altri motivi per i quali non dovrebbe essere usato, se invece ci sono altri pacchetti che dovrebbero essere utilizzati, esso dovrebbe essere indicato.
- 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 Home page originale del pacchetto

Si consiglia di aggiungere l'URL per la home page del pacchetto nel campo `Homepage` della sezione `Source` in `debian/control`. L'aggiunta di questa informazione nella descrizione del stesso pacchetto è considerata obsoleta.

6.2.6 La posizione del Version Control System

Ci sono campi aggiuntivi per la posizione del Version Control System in `debian/control`.

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.

L'informazione è destinata ad essere utile per l'utente finale, disposto a sfogliare l'ultimo lavoro svolto sul pacchetto (ad esempio quando si cerca la patch di un bug etichettato come `pending` nel sistema di bug tracking).

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).

L'informazione è destinata ad essere utile per un utente esperto nel dato Version Control System e disposto a compilare la versione attuale del pacchetto dai sorgenti VCS. Altri usi di queste informazioni possono includere compilazioni automatiche della versione VCS più recente del dato pacchetto. A tal fine la posizione a cui punta il campo dovrebbe essere la versione migliore rispetto a quella agnostica e puntare al ramo principale (per i VCS che supportano tale concetto). Inoltre, la posizione indicata dovrebbe essere accessibile per l'utente finale; soddisfare questo requisito potrebbe implicare un accesso anonimo al repository invece di puntare a una versione SSH-accessibile dello stesso.

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 Buone pratiche per il `debian/changelog`

Le seguenti pratiche integrano la Policy sui file changelog.

6.3.1 Scrivere informazioni utili nel file changelog

La voce del changelog inerente una revisione del pacchetto documenta i cambiamenti in quella specifica revisione e solo loro. Concentrarsi sulla descrizione di cambiamenti significativi e visibili all'utente che sono stati fatti dopo l'ultima versione.

Ci si concentri su *ciò* che è stato cambiato: chi, come e quando di solito sono meno importanti. Detto questo, ricordarsi di citare le persone che hanno fornito un notevole aiuto nel compilare il pacchetto (ad esempio, coloro che hanno inviato delle patch).

Non c'è bisogno di elaborare le modifiche banali e ovvie. È inoltre possibile aggregare diversi cambiamenti in un'unica voce. D'altra parte, non si sia troppo ermetici se si ha intrapreso un cambiamento importante. Si sia particolarmente chiari se ci sono cambiamenti che influenzano il comportamento del programma. Per ulteriori chiarimenti, utilizzare il `README.Debian`.

Si utilizzi l'inglese comune in modo che la maggior parte dei lettori possa comprenderlo. Si evitino abbreviazioni, termini tecnici e gergo quando si spiegano i cambiamenti che chiudono i bug, soprattutto per i bug segnalati dagli utenti che non sembrano particolarmente smaliziati dal punto di vista tecnico. Si sia gentili, non si imprechi.

A volte è desiderabile far precedere le voci del changelog con i nomi dei file che sono stati modificati. Tuttavia, non c'è bisogno di elencare esplicitamente uno per uno tutti i file modificati, soprattutto se il cambiamento è stato piccolo o ripetitivo. È possibile utilizzare i metacaratteri.

Quando si parla di bug, non si dia per scontato nulla. Dire quale era il problema, come è stato risolto e aggiungere in fondo la stringa closes: #nnnnn. Per ulteriori informazioni, si consulti *Quando i bug vengono chiusi da nuovi upload*.

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 *Aggiornamenti da unstable*). 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 Comuni incomprensioni sulle voci del changelog

Le voci del changelog **non** dovrebbero documentare generici problemi di packaging (Ehi, se si è alla ricerca di foo.conf, è in /etc/blah/.), dal momento che si suppone che gli amministratori e gli utenti siano a conoscenza di come queste cose sono generalmente gestite sui sistemi Debian. Parlatene, tuttavia, se si modifica la posizione di un file di configurazione.

Gli unici bug chiusi con una voce nel changelog dovrebbero essere quelli che sono effettivamente chiusi nella stessa versione del pacchetto. Chiudere bug non correlati nel changelog è cattiva pratica. Si veda *Quando i bug vengono chiusi da nuovi upload*.

Le voci del changelog **non** dovrebbero essere utilizzate per discussioni casuali con chi ha segnalato il bug (non vedo segmentation fault quando avvio foo con l'opzione bar; invia più informazioni al riguardo), dichiarazioni generali sulla vita, l'universo e tutto il resto (scusate questo caricamento mi ha preso così tanto tempo, ma ho preso l'influenza), o richieste di aiuto (la lista di bug su questo pacchetto è enorme, vi prego di dare una mano). Queste cose di solito non vengono notate, ma possono infastidire le persone che desiderano leggere le informazioni sulle modifiche effettive nel pacchetto. Per ulteriori informazioni su come utilizzare il sistema di bug tracking vedere *Rispondere ai bug*.

Si tratta di una vecchia tradizione per riconoscere bug corretti in un caricamento di un non-maintainer nella prima voce del changelog dell'appropriato maintainer. Siccome ora abbiamo il version tracking, è sufficiente mantenere le voci del changelog NMUed e citare questo fatto nella propria voce del changelog.

6.3.4 Errori frequenti nelle voci del changelog

I seguenti esempi dimostrano alcuni errori comuni o di cattivo stile nelle voci del changelog.

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

Questo non dice ai lettori qualcosa di particolarmente utile, ovviamente.

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

Qual'era l'argomento della patch?

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

Revisione che ha completato cosa? L'ipotetica citazione notturna è lì a ricordarci che non dovremmo fidarci di quel codice?

```
* 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.
```

Innanzitutto, non c'è assolutamente alcun bisogno di caricare il pacchetto per trasmettere queste informazioni; invece, utilizzare il sistema di bug tracking. In secondo luogo, non c'è alcuna spiegazione del perché il rapporto non è un bug.

```
* 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
```

Dov'è la descrizione? Se non è possibile pensare ad un messaggio descrittivo, iniziare inserendo il titolo di ogni differente bug.

6.3.5 Integrare i changelog con i file NEWS.Debian

Importanti novità sui i cambiamenti in un pacchetto possono anche essere inserite nei file NEWS.Debian. Le notizie saranno visualizzate da strumenti come apt-listchanges, prima di tutto il resto dei changelog. Questo è il mezzo preferito per permettere all'utente di conoscere i cambiamenti significativi in ??un pacchetto. È meglio che usare le note di debconf in quanto è meno fastidioso e l'utente può tornare indietro e vedere il file NEWS.Debian dopo l'installazione. Ed è meglio rispetto all'elencare i principali cambiamenti presenti in README.Debian, dal momento che l'utente può facilmente perderli.

Il formato del file è lo stesso di un file changelog Debian, ma lasciare fuori gli asterischi e descrivere ogni notizia con un paragrafo completo quando necessario, piuttosto che le più concise sintesi che andrebbero in un changelog. È una buona idea eseguire il file attraverso dpkg-parsechangelog per controllare la formattazione in quanto durante la fase di compilazione non sarà controllata automaticamente come è stato fatto per il changelog. Ecco un esempio di un vero e proprio file NEWS.Debian:

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

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

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

Il file NEWS.Debian è installato come /usr/share/doc/*package*/NEWS.Debian.gz. È compresso e ha sempre quel nome, anche in pacchetti nativi Debian. Se si utilizza debhelper, dh_installchangelogs installerà il file debian/NEWS per voi.

A differenza dei file changelog, non è necessario aggiornare il file NEWS.Debian ad ogni rilascio. Aggiornarli solo se si ha qualcosa particolarmente degna di nota che l'utente dovrebbe conoscere. Se non si ha alcuna notizia, non c'è bisogno di fornire un file NEWS.Debian. Nessuna notizia è una buona notizia!

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 Buone pratiche per gli script del mantainer

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

Gli script del maintainer devono essere idempotenti. Ciò significa che è necessario assicurarsi che nulla di male accadrà se lo script dovesse essere invocato due volte dove di solito viene lanciato una volta sola.

Gli standard input e output possono essere reindirizzati (ad esempio nelle pipe) per finalità di logging, quindi non utilizzateli come una tty.

Tutte le configurazioni suggerite o interattive devono essere ridotte al minimo. Quando è necessario, si dovrebbe utilizzare il pacchetto debconf per l'interfaccia. Ricordare che il suggerimento in ogni caso può esserci solo nella fase configure dello script postinst.

Mantenere gli script del maintainer più semplici possibile. Si consiglia di utilizzare puri script POSIX. Ricordate, se si ha bisogno di tutte le funzioni di bash, lo script del maintainer deve avere una linea shebang per bash. La

shell POSIX o Bash sono preferite a quella Perl, poiché permettono a `debhelper` di aggiungere facilmente bit agli script.

Se si modificano gli script del maintainer, assicurarsi di testare la rimozione del pacchetto, la doppia installazione e l'epurazione. Assicurarsi che un pacchetto epurato sia completamente sparito, ovvero, deve rimuovere tutti i file creati, direttamente o indirettamente, in tutti gli script del maintainer.

Se è necessario verificare l'esistenza di un comando, si dovrebbe usare qualcosa simile a

```
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 Gestione della configurazione con `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.

Queste linee guida comprendono un certo stile di scrittura e di raccomandazioni tipografiche, considerazioni generali sull'uso di debconf e raccomandazioni più specifiche per alcune parti della distribuzione (il sistema di installazione, per esempio).

6.6.1 Non abusare di debconf

Da quando debconf è apparso in Debian, è stato ampiamente abusato e diverse critiche ricevute dalla distribuzione Debian provengono dall'abuso di debconf con la necessità di rispondere ad una vasta serie di domande prima di installare ogni piccola cosa.

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 Raccomandazioni generali per autori e traduttori

6.6.2.1 Scrivere inglese corretto

La maggior parte dei maintainer dei pacchetti Debian non sono di madrelingua inglese. Quindi, la scrittura di modelli correttamente formulati, può non essere facile per loro.

utilizzare (e abusare) della mailing list `debian-l10n-english@lists.debian.org`. Avrete le vostre bozze di modelli corrette.

I modelli scritti male danno una cattiva immagine del pacchetto, del proprio lavoro... o anche di Debian stesso.

Evitare il gergo tecnico il più possibile. Se alcuni termini suonano comuni a voi, possono essere impossibili da comprendere per gli altri. Se non si possono evitare, cercare di spiegarli (utilizzare la descrizione estesa). Nel fare ciò, cercare di bilanciarsi tra verbosità e semplicità.

6.6.2.2 Sii gentile con i traduttori

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.

Utilizzare i modelli basati su gettext. Installare `po-debconf` sul proprio sistema di sviluppo e leggete la sua documentazione (`man po-debconf` è un buon inizio).

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`.

Se si ha intenzione di modificare i modelli originali, utilizzare il sistema di notifica fornito con il pacchetto `po-debconf`, vale a dire il `podebconf-report-po`, per contattare i traduttori. I Traduttori più attivi sono molto reattivi e ottenere il loro lavoro incluso insieme con i vostri modelli modificati vi risparmierà caricamenti aggiuntivi. Se si utilizzano modelli basati su gettext, il nome del traduttore e gli indirizzi di posta elettronica sono menzionati negli header dei file PO e saranno utilizzati da `podebconf-report-po`.

Un uso consigliato di ciò che questa utility è:

```
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`).

Dare una scadenza ai traduttori è sempre apprezzato, in modo tale che possano organizzare il loro lavoro. Ricordare che alcuni gruppi di traduzione hanno un processo formalizzato di traduzione/revisione e un ritardo inferiore a 10 giorni è considerato irragionevole. Un ritardo più breve mette troppa pressione sul team di traduzione e dovrebbe essere utilizzato per modifiche di minore entità.

In caso di dubbio, si può anche contattare il team di traduzione per una determinata lingua (debian-l10n-xxxxx@lists.debian.org), o la mailing list `debian-i18n@lists.debian.org`.

6.6.2.3 Ordinare intere traduzioni quando si correggono errori di battitura e di ortografia

Quando il testo di un modello debconf è corretto e si è **sicuri** che la modifica **non** influenzi le traduzioni, si sia gentili con i traduttori e *sistemare* le loro traduzioni.

Se non si fa così, l'intero modello non verrà tradotto fino a quando un traduttore invierà un aggiornamento.

Per *sistemare* le traduzioni, è possibile utilizzare `msguntypot` (parte del pacchetto `po4a`).

1. Rigenerare i file POT e PO.

   ```
   debconf-updatepo
   ```

2. Creare una copia del file POT.

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

3. Fare una copia di tutti i files PO.

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

4. Modificare i file dei modelli di debconf per correggere errori di battitura.
5. Rigenerare i file POT e PO (di nuovo).

```
debconf-updatepo
```

   A questo punto, la correzione dell'errore di battitura ha confuso tutte le traduzioni e questo spiacevole cambiamento è l'unico tra i file PO della vostra cartella principale e quella in frigo. Ecco come risolvere questa situazione.

6. Scartare la traduzione disordinata, ripristinare quella proveniente dal frigo.

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

7. Unire manualmente i files PO con il nuovo file POT, ma prendendo nell'account l'inutile disordine.

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

8. Pulizia.

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

6.6.2.4 Non fare ipotesi sulle interfacce

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.

I modelli di frasi dovrebbero anche evitare di menzionare i valori predefiniti nella loro descrizione. Primo, perché questo è ridondante con i valori visualizzati dagli utenti. Inoltre, perché questi valori predefiniti possono essere diversi dalle scelte del maintainer (per esempio, quando il database debconf è stato preconfigurato).

Più in generale, cercare di evitare di riferirsi alle azioni dell'utente. Basta indicare i fatti.

6.6.2.5 Non utilizzare la prima persona

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 Usare il genere neutro

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 Definizione dei campi dei modelli

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

6.6.3.1 Tipo

stringa

Risultati in un campo di input nel quale l'utente può digitare qualsiasi frase.

password

Richiede all'utente una password. La si usi con cautela; si sia consapevoli del fatto che la password che l'utente inserisce sarà scritta nel database di debconf. Si dovrebbe cancellare quel valore fuori dal database appena è possibile.

booleano

Una scelta vero/falso. Ricordare: vero/falso, **non si/no**...

seleziona

Una scelta tra una serie di valori. Le scelte devono essere specificate in un campo denominato «Choices». Separare i possibili valori con le virgole e gli spazi, in questo modo: `Scelte: sì, no, forse`.

Se le scelte sono stringhe traducibili, il campo «Choices» può essere contrassegnato come traducibile utilizzando `__Choices`. La doppia sottolineatura suddividerà ogni scelta in una stringa separata.

Il sistema `po-debconf` offre anche interessanti possibilità di marcare solamente **alcune** scelte come traducibili. Esempio:

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

In questo esempio, solo la stringa «Other» è traducibile, mentre altri sono acronimi che non devono essere tradotti. Quanto sopra esposto consente solo ad «Other» di essere inserito nei file PO e POT.

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).

nota

Piuttosto che essere una questione di per sé, questo tipo di dato indica una nota che può essere visualizzata all'utente. Dovrebbe essere usato solo per le note importanti che l'utente in realtà dovrebbe vedere, dato che debconf andrà incontro a grandi dolori per assicurarsi che l'utente la veda; arrestando l'installazione per consentire loro di premere un tasto persino inviandogli la nota tramite posta elettronica in alcuni casi.

testo

Questo tipo è ora considerato obsoleto: non usarlo.

errore

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).

Si consiglia di utilizzare questo tipo per ogni messaggio che necessita dell'attenzione dell'utente per una correzione di qualsiasi tipo.

6.6.3.2 Descrizione: descrizione breve ed estesa

Le descrizioni dei modelli constano di due parti: breve ed estesa. La descrizione breve è nella linea «Description:» del modello.

La descrizione breve dovrebbe essere mantenuta breve (50 caratteri o giù di lì) in modo che possa essere accolta dalla maggior parte delle interfacce di debconf. Mantenerla breve aiuta anche i traduttori, considerato che normalmente le traduzioni tendono a finire per essere più lunghe rispetto all'originale.

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?"

La descrizione breve non deve necessariamente essere un periodo completo. Questo fa parte della raccomandazione di mantenerla breve ed efficiente.

La descrizione estesa non deve ripetere la descrizione breve parola per parola. Se non è possibile pensare ad una descrizione lunga, quindi primo, si pensi un po' di più. Si condivida in debian-devel. Si chieda aiuto. Ci si iscriva ad un corso di scrittura! La descrizione estesa è importante. Se dopo tutto questo non è ancora possibile trovare qualcosa, la si lasci vuota.

La descrizione estesa dovrebbe usare periodi completi. I paragrafi devono essere brevi per migliorare la leggibilità. Non mescolare due idee nello stesso punto, piuttosto si usi un altro paragrafo.

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.

La descrizione estesa non dovrebbe **mai** includere una domanda.

Per specifiche regole inerenti il tipo di template (string, boolean, etc), leggere qui di seguito.

6.6.3.3 Scelte

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

Questo campo è facoltativo. Esso contiene la risposta predefinita per i modelli di periodi, di selezione singola e multipla. Per i modelli di selezione multipla, può contenere un elenco di scelte separate da virgole.

6.6.4 Template fields specific style guide

6.6.4.1 Type field

Nessuna indicazione specifica eccetto: utilizzare il tipo appropriato, facendo riferimento alla sezione precedente.

6.6.4.2 Il campo Description

Qui di seguito sono istruzioni specifiche per scrivere correttamente la descrizione (breve e lunga) a seconda del tipo di modello.

Modelli di string/password

- La descrizione breve è un suggerimento e **non** un titolo. Evitare domande in stile suggerimento (indirizzo IP?) a favore di suggerimenti aperti (indirizzo IP:). Si consiglia l'uso dei due punti.
- La descrizione estesa è un complemento della descrizione breve. Nella parte estesa, si spieghi ciò che viene chiesto, piuttosto che riproporre la stessa domanda con parole più lunghe. Utilizzare frasi complete. Una scrittura concisa è fortemente sconsigliata.

Modelli booleani

- 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).
- Anche in questo caso, evitare il riferimento a specifici widget delle interfacce. Un errore comune per tali modelli è se si risponde con costrutti di tipo Yes.

Selezione/Multiselezione

- 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... :)
- La descrizione estesa completerà la descrizione breve. Può far riferimento alle scelte disponibili. Può anche indicare che l'utente può scegliere più di una tra le scelte disponibili, se il modello è di tipo selezione multipla (l'interfaccia spesso rende chiaro tutto ciò).

Notes

- La descrizione breve dovrebbe essere considerata un **titolo**.
- La descrizione estesa è ciò che sarà visualizzato come una spiegazione più dettagliata della nota. Frasi, non scrittura sintetica.
- **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 Campo Choises

Se le «Choises» sono suscettibili di cambiare spesso, considerare l'uso del trucco «__ Choices». Questo dividerà ogni singola scelta in una singola stringa, che aiuterà notevolmente i traduttori nel compiere il loro lavoro.

6.6.4.4 Campo 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 *Sii gentile con i traduttori*), consider making this field translatable, if you think it may be translated.

Esempio, tratto dai modelli del pacchetto 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 Internazionalizzazione

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 Gestione delle traduzioni debconf

Come gli autori di port, i traduttori hanno un compito difficile. Lavorano su molti pacchetti e devono collaborare con molti maintainer diversi. Inoltre, la maggior parte delle volte, non sono di madrelingua inglese, quindi si potrebbe dover essere particolarmente pazienti con loro.

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 Documentazione internazionalizzata

Internazionalizzare la documentazione è fondamentale per gli utenti, ma richiede molto lavoro. Non c'è modo di eliminare tutto quel lavoro, ma si possono rendere le cose più facili per i traduttori.

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 Situazioni comuni durante la pacchettizzazione

6.9.1 Pacchettizzazione utilizzando `autoconf/automake`

Mantenere i file `config.sub` e `config.guess` di `autoconf` aggiornati è fondamentale per gli autori di port, soprattutto su architetture più volatili. Alcune ottime pratiche di packaging per ogni pacchetto che usa `autoconf` e/o `automake` sono state sintetizzate in `/usr/share/doc/autotools-dev/README.Debian.gz` dal pacchetto `autotools-dev`. Si è vivamente incoraggiati a leggere questo file e di seguire le raccomandazioni in esso fornite.

6.9.2 Le librerie

Le librerie sono sempre difficili da pacchettizzare per vari motivi. La policy impone molti vincoli per facilitare il loro mantenimento e per assicurarsi che gli aggiornamenti siano i più semplici possibili quando una nuova versione viene pubblicata. Una rottura in una libreria può causare una rottura in decine di pacchetti dipendenti.

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

6.9.3 La documentazione

Assicurarsi di seguire la policy Policy sulla documentazione.

Se il pacchetto contiene la documentazione compilata a partire da XML o SGML, si consiglia di non includere il sorgente XML o SGML nel pacchetto binario. Se gli utenti vogliono il sorgente della documentazione, dovrebbero poter recuperare il pacchetto sorgente.

La policy specifica che la documentazione deve essere fornita in formato HTML. Si consiglia anche di inserire la documentazione in formato PDF e testo normale se conveniente e se è possibile output di discreta qualità. Tuttavia, non è in genere appropriato includere la versione in testo semplice della documentazione il cui formato sorgente è HTML.

La maggior parte dei manuali dovrebbe registrarsi con `doc-base` durante l'installazione. Si consulti la documentazione del pacchetto `doc-base` per ulteriori informazioni.

La policy di Debian (sezione 12.1) indica che le pagine del manuale dovrebbero accompagnare ogni programma, utility e la funzione e suggerirli per altri oggetti come file di configurazione. Se il lavoro che si sta pacchettizzando non ha queste pagine di manuale, si consideri la loro scrittura per l'inclusione nel pacchetto e di sottoporla allo sviluppatore originale.

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 Specifici tipi di pacchetti

Vari tipi specifici di pacchetti hanno particolari sub-politiche e rispettive pratiche e regole per la pacchettizzazione:

- 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.
- I relativi pacchetti Emacs hanno la emacs policy.
- I relativi pacchetti Java hanno la loro java policy.
- 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.
- I pacchetti che forniscono XML o SGML DTD devono essere conformi alle indicazioni fornite nel pacchetto `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 Dati indipendenti dall'architettura

Non è raro avere una grande quantità di dati indipendenti dall'architettura pacchettizzati con un programma. Ad esempio, i file audio, una collezione di icone, modelli di wallpaper, o altri file grafici. Se la dimensione di questi dati è trascurabile rispetto alle dimensioni del resto del pacchetto, probabilmente è meglio tenere il tutto in un unico pacchetto.

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 *Strumenti per la pulizia di pacchetti*) when run over the entire Debian archive.

6.9.6 Aver bisogno di una particolare localizzazione durante la compilazione

Se si ha bisogno di una certa localizzazione durante la compilazione, si può creare un file temporaneo con questo trucco:

Se si imposta `LOCPATH` all'equivalente di `/usr/lib/locale` e `LC_ALL` al nome del locale che si genera, si dovrebbe ottenere ciò che si desidera senza essere root. Qualcosa di simile a questo:

```
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 Rendere i pacchetti di transizione conformi a deborphan

Deborphan è un programma per aiutare gli utenti ad individuare quali pacchetti possono essere tranquillamente rimossi dal sistema, vale a dire quelli che non hanno pacchetti dipendenti da loro. Il funzionamento predefinito è di cercare solo all'interno delle sezioni libs e oldlibs, per dare la caccia a librerie inutilizzate. Ma quando viene passato l'argomento giusto, cerca di catturare altri pacchetti inutili.

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 Buone pratiche per i file `.orig.tar.{gz,bz2,xz}`

Ci sono due tipi di tarball dei sorgenti originali: sorgente puro e sorgente originale ripacchettizzato.

6.9.8.1 Sorgente puro

La caratteristica distintiva di un tarball sorgente incontaminato è che il `.orig.tar.{gz,bz2,xz}` è byte-per-byte identico a un tarball ufficialmente distribuito dall'autore originale.[1] Questo rende possibile l'utilizzo di checksum per verificare facilmente che tutte le modifiche tra la versione di Debian e quella originale siano contenute nel Debian diff. Inoltre, se il sorgente originale è enorme, gli autori originali e chiunque possieda già l'archivio originale possono risparmiare tempo di download, se vogliono ispezionare il pacchetto in dettaglio.

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. Si scompatta l'archivio in una cartella temporanea vuota facendo

[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.

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

2. Se, dopo questo, la cartella temporanea non contiene nulla, ma una sola cartella e nessun altro file, `dpkg-source` rinomina quella cartella in *packagename-upstream-version*`(.orig)`. Il nome della più alta cartella nella tarball non ha importanza, ed è tralasciata.

3. In caso contrario, l'archivio originale deve essere stato confezionato senza una comune cartella di livello alto (vergogna sull'autore originale!). In questo caso, `dpkg-source` rinomina la cartella temporanea *stessa* in *packagename-upstream-version*`(.orig)`.

6.9.8.2 Sorgente originale ripacchettizzato

Si dovrebbe caricare i pacchetti con un tarball sorgente puro, se possibile, ma ci sono vari motivi per cui potrebbe non essere possibile. Questo è il caso in cui se l'autore originale non distribuisce il sorgente come non completamente compresso in formato tar, o se il tarball originale contiene materiale non DFSG-free che è necessario rimuovere prima di caricare.

In questi casi, lo sviluppatore deve costruirsi un `.orig.tar.{gz,bz2,xz}` adatto. Ci si riferisce a un tale tarball come sorgente originale ripacchettizzato. Si noti che un sorgente originale ripacchettizzato è diverso da un pacchetto Debian nativo. Un sorgente originale ripacchettizzato con modifiche specifiche per Debian ancora viene distribuito in un separato `.diff.gz` o `.debian.tar.{gz,bz2,xz}` e ha ancora un numero di versione composto da *upstream-version* e *debian-version*.

Ci possono essere casi in cui è auspicabile pacchettizzare nuovamente il sorgente anche se l'autore originale distribuisce un `.tar.{gz,bz2,xz}` che potrebbe in linea di principio essere utilizzato nella sua forma originaria. Il più ovvio è se un *significativo* risparmio di spazio può essere ottenuto ricomprimendo l'archivio tar o rimuovendo genuinamente dell'inutile fuffa dall'archivio originale. Si usi la propria discrezione qui, ma si sia pronti a difendere la propria decisione se si pacchettizza nuovamente il sorgente che poteva esser puro.

Un `.orig.tar.{gz,bz2,xz}` ripacchettizzato

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.

2. **non dovrebbe** contenere qualsiasi tipo di file che non proviene dall'autore originale, o il cui contenuto è stato modificato.[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.

   (*Motivazione:* È comune per gli utenti Debian che hanno bisogno di compilare software per piattaforme non-Debian prendere il sorgente da uno dei mirror Debian piuttosto che cercare di individuare un punto canonico di distribuzione originale).

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.

[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.

6.9.8.3 Modifica dei file binari

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 `uuencoded` (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 Buone pratiche per i pacchetti di debug

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.

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.

Si noti che il pacchetto di debug dovrebbe dipendere dal pacchetto per il quale fornisce i simboli di debug e questa dipendenza dovrebbe essere versionata. Per esempio:

```
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 Buone pratiche per i metapacchetti

Un metapacchetto è un pacchetto per lo più vuoto che rende facile installare un insieme coerente di pacchetti che possono evolvere nel tempo. Realizza ciò creando una dipendenza per tutti i pacchetti dell'insieme. Grazie alla potenza di APT, il maintainer del metapacchetto può sistemare le dipendenze e il sistema dell'utente otterrà automaticamente i pacchetti supplementari. I pacchetti eliminati che sono stati installati automaticamente verranno contrassegnati come candidati alla rimozione (e saranno anche rimossi automaticamente da aptitude). gnome e linux-image-amd64 sono due esempi di metapacchetti (compilati dai pacchetti sorgente meta-gnome2 e 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.

CAPITOLO 7

Oltre la pacchettizzazione

Debian è molto più di un semplice software di confezionamento e di mantenimento di questi pacchetti. Questo capitolo contiene informazioni sui modi, modi spesso molto critici, per contribuire a Debian oltre la semplice creazione e la manutenzione dei pacchetti.

Come organizzazione di volontariato, Debian si basa sulla discrezione dei suoi membri nella scelta di ciò su cui vogliono lavorare e nello scegliere la cosa più critica sulla quale trascorre la maggior parte del proprio tempo.

7.1 Segnalare i bug

Si consiglia di notificare i bug appena li si trovi nei pacchetti Debian. In realtà, gli sviluppatori Debian sono spesso la prima linea di tester. L'individuazione e la segnalazione di bug nei pacchetti di altri sviluppatori migliora la qualità di Debian.

Si leggano le istruzioni per la segnalazione di bug nel sistema di tracciamento dei bug di Debian.

Provare a presentare il bug da un account di un utente normale dove si preferisce ricevere la posta, in modo che le persone possano rintracciarvi se hanno bisogno di ulteriori informazioni sul bug. Non inviare bug come root.

È possibile utilizzare uno strumento come reportbug 1 per segnalare bug. È in grado di automatizzare e in generale facilitare il processo.

Assicurarsi che il bug non sia già stato presentato per un pacchetto. Ogni pacchetto ha una lista di bug facilmente raggiungibile a `http://bugs.debian.org/`*nomepacchetto*. Programmi di utilità come querybts 1 possono anche fornire queste informazioni (e anche `reportbug` di solito invocherà querybts prima di inviare).

Cercare di indirizzare i bug nella posizione corretta. Quando per esempio il proprio bug è relativo ad un pacchetto che sovrascrive i file appartenenti ad un altro pacchetto, controllare le liste di bug per *entrambi* questi pacchetti in modo da evitare di presentare duplicati di segnalazioni di bug.

Per maggior credito, si può passare attraverso altri pacchetti, unificando i bug che sono riportati più di una volta, o etichettando bug «fixed» quando sono già stati risolti. Si noti che quando non si è né il mittente del bug né il maintainer del pacchetto, non si dovrebbe in realtà chiudere il bug (a meno che non si ha il permesso del maintainer).

Di tanto in tanto si consiglia di controllare lo stato di avanzamento del bug che si è inviato. Cogliere l'occasione di chiudere quelli che non è possibile più riprodurre. Per scoprire tutti i bug che si sono inviati, è sufficiente visitare `http://bugs.debian.org/from:`*proprio-indirizzo-email*.

7.1.1 Riportare molti bug in una sola volta (presentazione massiccia di bug)

Segnalare un gran numero di bug per lo stesso problema su un gran numero di pacchetti differenti - vale a dire, più di 10 - è una pratica sconsigliata. Prendete tutte le misure possibili per evitare del tutto di sottoporre bug massicci. Per esempio, se il controllo per il problema può essere automatizzato, aggiungere un nuovo controllo per `lintian` in modo che venga emesso un errore o un avviso.

Se si riportano più di 10 bug sullo stesso argomento in una sola volta, si consiglia di inviare un messaggio a `debian-devel@lists.debian.org` dove descrivete la vostra intenzione prima di presentare il rapporto e di menzionarla nell'oggetto della mail. Questo permetterà ad altri sviluppatori di verificare che il bug sia un problema reale. Inoltre, aiuterà a prevenire una situazione in cui molti maintainer iniziano ad occuparsi simultaneamente dello stesso bug.

Utilizzare i programmi `dd-list` e se appropriato `whodepends` (dal pacchetto `devscripts`) per generare un elenco di tutti i pacchetti interessati, ed includere l'output nella propria email indirizzata a `debian-devel@lists.debian.org`.

Si noti che quando si inviano molti bug sullo stesso argomento, si dovrebbe inviare la segnalazione di bug a `maintonly@bugs.debian.org` in modo che la segnalazione non venga inoltrata alla mailing list di distribuzione bug.

The program `mass-bug` (from the package `devscripts`) can be used to file bug reports against a list of packages.

7.1.1.1 Usertag

Si potrebbe voler utilizzare le usertag BTS al momento di presentare i bug per un certo numero di pacchetti. Le usertag sono simili alle normali tag come «patch» e «wishlist», ma differiscono nel senso che sono definite dall'utente e occupano uno spazio dei nomi univoco per un particolare utente. Questo consente a più insiemi di sviluppatori di «usertaggare» lo stesso bug in diversi modi senza conflitto.

Per aggiungere usertag al momento della presentazione di bug, specificare gli pseudo-header `User` e `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.

Per visualizzare bug etichettati con uno specifico usertag, visitare la pagina `http://bugs.debian.org/cgi-bin/pkgreport.cgi?users=`*email-addr*`&tag=`*tag-name*.

7.2 Costo della Quality Assurance

7.2.1 Lavoro giornaliero

Anche se c'è un gruppo dedicato di persone per la Quality Assurance, le mansioni QA non sono riservate esclusivamente a loro. È possibile partecipare a questo sforzo, mantenendo i vostri pacchetti il più possibile privi di bug e il più possibile lintian-clean (si consulti *lintian*). Se non si riesce a farlo, allora si dovrebbe prendere in considerazione di rendere orfani alcuni dei vostri pacchetti (si consulti *Pacchetto orfano*). In alternativa, si può chiedere l'aiuto di altre persone al fine di recuperare il ritardo con i bug arretrati che si hanno (si può chiedere aiuto su `debian-qa@lists.debian.org` o `debian-devel@lists.debian.org`). Allo stesso tempo, si può cercare un co-maintainer (si consulti *La manutenzione collaborativa*).

7.2.2 Bug squashing parties

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.

Se non ci si sente sicuri di fare un NMU, basta inviare una patch al BTS. È molto meglio di un NMU non funzionante.

7.3 Come contattare gli altri maintainer

Durante il corso della vita all'interno di Debian, si dovranno contattare altri maintainer per vari motivi. Si vorrà discutere di un nuovo modo di cooperare tra un insieme di pacchetti correlati, o semplicemente si vorrà ricordare a qualcuno che una nuova versione è disponibile e che se ne ha bisogno.

Cercare l'indirizzo email del maintainer del pacchetto può essere fonte di distrazione. Fortunatamente, c'è un semplice alias di posta elettronica, *package*`@packages.debian.org`, che fornisce un modo per inviare email al maintainer, qualunque possa essere il loro indirizzo di posta elettronica individuale (o gli indirizzi). Sostituite *package* con il nome di un sorgente o un pacchetto binario.

Si potrebbe anche essere interessati a contattare le persone che sono iscritte ad un determinato pacchetto sorgente tramite *The Debian Package Tracker*. È possibile farlo utilizzando l'indirizzo email *package*`@packages.qa.debian.org`.

7.4 Rapportarsi con maintainer non attivi e/o non raggiungibili

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.

Il primo passo è quello di contattare educatamente il maintainer, ed attendere un ragionevole lasso di tempo per la risposta. È abbastanza difficile definire tempo ragionevole, ma è importante tenere in considerazione che la vita reale a volte è molto frenetica. Un modo per gestire questa situazione sarebbe quello di inviare un sollecito dopo due settimane.

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.

Se il maintainer non risponde entro quattro settimane (un mese), si può supporre che la risposta probabilmente non arriverà. Se ciò accade, si dovrebbe indagare ulteriormente e cercare di raccogliere quante più informazioni utili possibili sul maintainer in questione. Ciò include:

- Le informazioni `echelon` disponibili attraverso il database LDAP degli sviluppatori, che indica quando lo sviluppatore ha pubblicato l'ultimo post in una mailing list Debian. (Questo include email su aggiornamenti distribuiti tramite la lista `debian-devel-changes@lists.debian.org`). Inoltre, ricordare di controllare nel database se il maintainer è contrassegnato come in vacanza.
- Il numero di pacchetti dei quali questo maintainer è responsabile e lo stato di quei pacchetti. In particolare, ci sono dei bug RC che sono stati aperti da tempo? Inoltre, quanti bug ci sono in generale? Un altro pezzo importante di informazioni è se i pacchetti sono stati NMUed e se sì, da chi.
- C'è qualche attività del maintainer al di fuori di Debian? Ad esempio, potrebbero aver inviato qualcosa di recente a mailing list non-Debian o newsgroup.

Un po' problematici sono i pacchetti che sono stati sponsorizzatiì: il maintainer non è uno sviluppatore Debian ufficiale. Le informazioni `echelon` non sono disponibili per le persone sponsorizzate, per esempio, quindi è necessario trovare e contattare lo sviluppatore Debian che ha effettivamente caricato il pacchetto. Dato che hanno firmato il pacchetto, che sono responsabili per il caricamento in ogni caso, e sono probabilmente intenzionati di sapere cosa è successo alla persona che hanno sponsorizzato.

È anche consentito inviare una query a `debian-devel@lists.debian.org` chiedendo se qualcuno è a conoscenza del luogo in cui si trova il maintainer mancante. Si metta in Cc: la persona in questione.

Dopo aver raccolto tutto questo, è possibile contattare `mia@qa.debian.org`. Persone su questo alias utilizzeranno le informazioni fornite al fine di decidere come procedere. Per esempio, potrebbero rendere orfano uno o tutti i pacchetti del maintainer. Se un pacchetto è stato NMUed, potrebbero preferire di contattare la NMUer prima di rendere orfano il pacchetto; forse la persona che ha fatto la NMU è interessato al pacchetto.

Un'ultima parola: ricordare di essere educati. Si è tutti volontari e non si può dedicare tutto il proprio tempo a Debian. Inoltre, non si è a conoscenza delle circostanze della persona che è coinvolta. Forse potrebbe essere gravemente malata o potrebbe anche essere morta: non potete sapere chi potrebbe essere dall'altra parte. Si immagini come un parente si sentirà se leggesse l'email del defunto e trovasse un messaggio molto scortese, arrabbiato e accusatorio!

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 Interagire con potenziali sviluppatori Debian

Il successo di Debian dipende dalla sua capacità di attrarre e trattenere nuovi e talentuosi volontari. Se si è uno sviluppatore esperto, si consiglia di partecipare al processo per coinvolgere nuovi sviluppatori. Questa sezione descrive come aiutare i nuovi potenziali sviluppatori.

7.6.1 Sponsorizzare pacchetti

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.

Gli sviluppatori Debian possono sponsorizzare i pacchetti. I maintainer Debian non possono.

Il processo di sponsorizzazione di un pacchetto è:

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. Lo sponsor esamina il pacchetto sorgente. Se trova dei problemi, informa il maintainer chiedendogli di fornire una versione corretta (il processo inizia dal punto 1).
4. Lo sponsor non può trovare tutti i problemi che ci sono. Compila il pacchetto, lo firma e lo carica su 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?

Ci si dovrebbe anche assicurare che il potenziale maintainer sarà un buon maintainer. Hanno già una certa esperienza con altri pacchetti? Se sì, stanno facendo un buon lavoro con loro (corretti alcuni bug)? Hanno familiarità con il pacchetto e con il suo linguaggio di programmazione? Hanno le competenze necessarie per questo pacchetto? In caso contrario, sono in grado di impararle?

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 Sponsorizzare un nuovo pacchetto

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.

Non bisogna mai sponsorizzare un nuovo pacchetto senza revisionarlo. La revisione dei nuovi pacchetti realizzata da ftpmasters assicura principalmente che il software sia veramente libero. Certo, capita che inciampino sui problemi di pacchettizzazione, ma in realtà non dovrebbero. È il proprio compito garantire che il pacchetto caricato sia conforme alle Free Software Guidelines di Debian e sia di buona qualità.

La compilazione del pacchetto e il testare il software è parte della revisione, ma non è anche sufficiente. Il resto di questa sezione contiene un elenco non esaustivo dei punti da controllare nella vostra revisione.[1]

- Verificare che il tarball originale fornito sia lo stesso che è stato distribuito dall'autore principale (quando i sorgenti sono stati ripacchettizzati per Debian, generare da soli la tarball modificata).
- 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.
- Compilare il pacchetto con `pbuilder` (o qualsiasi strumento simile, vedi *pbuilder*) per garantire che le dipendenze di compilazione siano soddisfatte.

[1] You can find more checks in the wiki, where several developers share their own sponsorship checklists.

- Correggere `debian/control`: segue le buone pratiche (si consulti *Buone pratiche per debian/control*)? Sono soddisfatte le dipendenze?
- Correggere `debian/rules`: rispecchia le buone pratiche (si consulti *Buone pratiche per debian/rules*)? Vedete alcuni possibili miglioramenti?
- Correggere gli script del maintainer (`preinst`, `postinst`, `prerm`, `postrm`, `config`): `preinst/postrm` funzioneranno quando le dipendenze non sono installate? Tutti gli script sono idempotenti (cioè si possono eseguire più volte senza conseguenze)?
- Controllare ogni modifica ai file originali (sia in `.diff.gz`, o in `debian/patches/` o direttamente incluse nella tarball `debian` dei file binari). Sono giustificate? Sono adeguatamente documentate (con DEP-3 per le patch)?
- Per ogni file, ci si chieda perché il file è lì e se è il modo giusto per ottenere il risultato desiderato. Il maintainer sta seguendo le best practices per la pacchettizzazione (si consulti *Buone pratiche per la pacchettizzazione*)?
- Build the packages, install them and try the software. Ensure that you can remove and purge the packages. Maybe test them with `piuparts`.

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 *The Debian Package Tracker*.

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
```

Se si usa `debuild` e `debsign`, si può anche configurarlo in modo permanente in `~/devscripts`.:

```
DEBSIGN_KEYID=KEY-ID
```

7.6.1.2 Sponsorizzare un aggiornamento di un pacchetto esistente

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`).

Se si confrontano i pacchetti sorgente (esclusi i file originali nel caso di una nuova versione, ad esempio filtrando l'output di `debdiff` con `filterdiff -i '*/debian/*'`), è necessario capire tutti i cambiamenti che vedrete e che devono essere adeguatamente documentati nel changelog 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 *The Debian Package Tracker*) 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.

Se non si è trovato alcun grande problema, caricare la nuova versione. In caso contrario, chiedere al maintainer di fornire una versione corretta.

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 Promuovere nuovi sviluppatori

Consultare la pagina Promuovere nuovi sviluppatori sul sito web di Debian.

7.6.4 Gestire nuove candidature di maintainer

Consultare la Checklist per i Gestori delle Candidature sul sito web di Debian.

CAPITOLO 8

Internazionalizzazione e traduzioni

Debian supporta un numero sempre crescente di lingue naturali. Anche se si è un madrelingua inglese e non si parla un'altra lingua, è parte del proprio dovere come maintainer essere a conoscenza delle problematiche di internazionalizzazione (i18n abbreviato perché ci sono 18 lettere tra la 'i' e la 'n' in internazionalizzazione). Pertanto, anche se non si hanno problemi con programmi solo inglesi, vi consigliamo di leggere la maggior parte di questo capitolo.

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 e i18n sono interconnessi, ma le difficoltà relative a ciascuno di essi sono molto diverse. Non è davvero difficile consentire ad un programma di cambiare la lingua in cui vengono visualizzati i testi sulla base di impostazioni utente, ma in realtà è molto dispendioso tradurre questi messaggi. D'altro canto, impostare la codifica dei caratteri è banale, ma adattare il codice per utilizzare diverse codifiche di carattere è un problema veramente difficile.

Lasciando da parte i problemi i18n, dove non è possibile dare alcuna linea guida generale, non vi è in realtà alcuna infrastruttura centrale per l10n in Debian che potrebbe essere confrontata con il meccanismo buildd per il porting. Così la maggior parte del lavoro deve essere fatto manualmente.

8.1 Come le traduzioni sono effettuate in Debian

La gestione della traduzione dei testi contenuti in un pacchetto è ancora un compito manuale e il processo dipende dal tipo di testo che si desidera vedere tradotto.

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.

Per le pagine web, ogni squadra l10n ha accesso alle VCS rilevanti, e sono disponibili presso il sito Central Debian translation statistics.

Per la documentazione di carattere generale su Debian, il processo è più o meno lo stesso che per le pagine web (i traduttori hanno accesso al VCS), ma non ci sono pagine di statistiche.

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 I18N e L10N FAQ per i maintainer

Questa è un elenco di problemi che i maintainer potrebbero dover affrontare in materia di i18n e l10n. Durante la lettura di questo documento, tenere presente che non esiste un vero consenso su questi punti all'interno di Debian e che questo è solo un consiglio. Se si ha un'idea migliore per un dato problema, o se si è in disaccordo su alcuni punti, non esitare a fornire il proprio feedback, in modo che questo documento possa essere migliorato.

8.2.1 Come ottenere un certo testo tradotto

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 Come ottenere una revisione di una data traduzione

Di volta in volta, gli individui traducono alcuni testi nel proprio pacchetto e vi chiederanno l'inserimento della traduzione nel pacchetto. Questo può diventare un problema se non si parla correntemente la lingua data. È una buona idea inviare il documento alla mailing list l10n corrispondente, chiedendo una revisione. Una volta che è stata fatta, ci si dovrebbe sentire più sicuri della qualità della traduzione e sentirsi sicuri da includerla nel proprio pacchetto.

8.2.3 Come ottenere una data traduzione aggiornata

Se si dispone di alcune traduzioni di un dato testo in giro, ogni volta che si aggiorna l'originale, si dovrebbe chiedere al traduttore precedente di aggiornare la traduzione con le nuove modifiche. Tenete a mente che questo compito richiede tempo, almeno una settimana per ottenere l'aggiornamento revisionato e tutto il resto.

Se il traduttore non risponde, si può chiedere aiuto sulla mailing list l10n corrispondente. Se tutto fallisce, non dimenticare di mettere un avviso nel documento tradotto, affermando che la traduzione è in qualche modo obsoleta e che il lettore dovrebbe fare riferimento al documento originale, se possibile.

Evitare di rimuovere una traduzione del tutto perché è obsoleta. La vecchia documentazione è spesso migliore di nessuna documentazione per chi non parla inglese.

8.2.4 Come gestire una segnalazione di bug riguardante una traduzione

La soluzione migliore potrebbe essere quella di marcare il bug come trasmesso al maintainer originale e trasmetterlo sia al traduttore precedente e alla loro squadra (utilizzando la mailing list debian-l10n-XXX corrispondente).

8.3 I18N & L10N FAQ per traduttori

Durante la lettura di questo documento, si tenga presente che non esiste una procedura generale all'interno di Debian relativa a questi punti, e che in ogni caso, si dovrebbe collaborare con il team e il maintainer del pacchetto.

8.3.1 Come aiutare lo sforzo di traduzione

Scegliere ciò che si desidera tradurre, assicurarsi che nessuno stia già lavorando su di esso (con la tua mailing list debian-l10n-XXX), tradurlo, chiedere la sua revisione da altri madrelingua sulla propria mailing list l10n e fornirlo al maintainer del pacchetto (si consulti il punto successivo).

8.3.2 Come fornire una traduzione per l' inclusione in un pacchetto

Assicurarsi che la traduzione sia corretta (chiedere la revisione sulla propria mailing list l10n) prima di fornirla per l'inclusione. Ciò consentirà di risparmiare tempo a tutti e di evitare il caos conseguente all'avere diverse versioni dello stesso documento nelle segnalazioni di bug.

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 L'attuale pratica consigliata riguardanti l'l10n

- Come maintainer, mai modificare le traduzioni in qualsiasi modo (anche di riformattare il layout), senza chiedere sulla mailing list l10n corrispondente. Si rischia ad esempio di rompere la codifica del file in questo modo. Inoltre, quello che si considera un errore può essere corretto (o addirittura necessario) nella lingua data.
- Come traduttore, se si trova un errore nel testo originale, assicurarsi di segnalarlo. I traduttori sono spesso i lettori più attenti di un dato testo, e se non segnalano gli errori che trovano, nessuno lo sarà.
- In ogni caso, si ricordi che il problema principale con l10n è che richiede a più persone di collaborare, e che è molto facile iniziare discussioni su piccoli problemi a causa di incomprensioni. Quindi, se si hanno problemi con il proprio interlocutore, si chieda aiuto sulla mailing list l10n corrispondente, su debian-i18n, o anche su debian-devel (ma attenzione, discussioni l10n molto spesso diventano flame su quella lista :)
- In ogni caso, la cooperazione può essere raggiunta solo con il **rispetto reciproco**.

CAPITOLO 9

Panoramica degli strumenti del Debian Maintainer

Questa sezione contiene una panoramica generale degli strumenti disponibili per i maintainer. Ciò che segue non è affatto completo o definitivo, ma solo una guida per alcuni degli strumenti più popolari.

Gli strumenti del maintainer Debian hanno lo scopo di aiutare gli sviluppatori e liberare il loro tempo per attività critiche. Come dice Larry Wall, c'è più di un modo per farlo.

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 Strumeti di base

I seguenti strumenti sono praticamente obbligatori per qualsiasi maintainer.

9.1.1 dpkg-dev

`dpkg-dev` contiene gli strumenti (compreso `dpkg-source`) necessari per spacchettare, creare e caricare pacchetti sorgenti Debian. Queste utilità contengono la fondamentale funzionalità di basso livello necessaria per creare e manipolare i pacchetti; in quanto tali, essi sono essenziali per qualsiasi maintainer Debian.

9.1.2 debconf

`debconf` fornisce un'interfaccia coerente per la configurazione di pacchetti interattiva.È indipendente dall'interfaccia utente, che consente agli utenti finali di configurare i pacchetti con un'interfaccia testuale, un'interfaccia HTML o un'interfaccia con riquadri di dialogo. Nuove interfacce possono essere aggiunte come moduli.

È possibile trovare la documentazione per questo pacchetto nel pacchetto `debconf-doc`.

Many feel that this system should be used for all packages that require interactive configuration; see *Gestione della configurazione con debconf*. `debconf` is not currently required by Debian Policy, but that may change in the future.

9.1.3 fakeroot

`fakeroot` simula i privilegi di root. Questo permette di compilare pacchetti senza essere root (i pacchetti di solito vogliono installare i file con i permessi di root). Se si ha `fakeroot` installato, è possibile compilare pacchetti come utente normale: `dpkg-buildpackage-rfakeroot`.

9.2 Strumenti per la pulizia di pacchetti

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` studia approfonditamente i pacchetti Debian e fornisce informazioni su bug e violazioni delle policy. Contiene controlli automatici per molti aspetti della policy Debian così come alcuni controlli per gli errori più frequenti.

Si dovrebbe ottenere periodicamente il più nuovo `lintian` da `unstable` e controllare tutti i propri pacchetti. Si noti che l'opzione `-i` fornisce spiegazioni dettagliate di ciò che ogni errore o avvertimento significa, quale sia la sua base nella Policy e come generalmente si può risolvere il problema.

Si consulti *Testare del pacchetto* per maggiori informazioni su come e quando utilizzare Lintian.

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` (dal pacchetto `devscripts`, *devscripts*) confronta elenchi di file e file di controllo di due pacchetti. Si tratta di un test di regressione semplice, in quanto aiuterà a notare se il numero di pacchetti binari è cambiato dall'ultima operazione di caricamento, o se qualcosa è cambiato nel file di controllo. Naturalmente, alcune delle modifiche che segnalerà saranno a posto, ma può aiutare a prevenire vari incidenti.

È possibile eseguirlo su un paio di pacchetti binari:

```
debdiff package_1-1_arch.deb package_2-1_arch.deb
```

O anche su un paio di file di modifiche:

```
debdiff package_1-1_arch.changes package_2-1_arch.changes
```

Per ulteriori informazioni consultare debdiff1.

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 Strumenti ausiliari per debian/rules

Gli strumenti per la compilazione dei pacchetti rendono il processo di scrittura dei file debian/rules più facile. Per ulteriori informazioni sul motivo per cui questi potrebbero o non potrebbero essere desiderati si consulti *Script di supporto*.

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.

Ci sono una serie di piccoli pacchetti aggiuntivi per debhelper, troppo effimeri da documentare. È possibile vedere l'elenco della maggior parte di loro invocando 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.

Mentre i file di rules generati da dh_make sono, in generale, una base sufficiente per un pacchetto funzionante, sono ancora solo le basi: il maintainer ha ancora l'onere per sintonizzare con precisione i file generati e rendere il pacchetto completamente funzionante e conforme alla Policy.

9.3.3 equivs

equivs è un altro pacchetto per fare i pacchetti. È spesso suggerito per uso locale, se si ha bisogno di creare un pacchetto semplicemente per soddisfare le dipendenze. A volte è anche utilizzato quando si costruiscono «meta-pacchetti», che sono pacchetti il cui unico scopo è quello di dipendere da altri pacchetti.

9.4 Strumenti di compilazione

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 fornisce la capacità di iniettare o importare pacchetti sorgenti Debian in un repository GIT, di compilare un pacchetto Debian dal repository GIT, ed aiuta a integrare i cambiamenti del codice originale nel repository.

Queste utilità forniscono un'infrastruttura per facilitare l'uso del GIT da parte del maintainer Debian. Questo permette di mantenere in GIT rami separati di un pacchetto per le distribuzioni stable, unstable e possibilmente experimental, con gli altri benefici di un sistema di controllo di versioni.

9.4.2 debootstrap

Il pacchetto e lo script `debootstrap` permettono di avviare un sistema Debian di base in qualsiasi parte del proprio filesystem. Per sistema di base, si intende il minimo dei pacchetti necessari al funzionamento e ad installare il resto del sistema.

Avere un sistema come questo può essere utile in molti modi. Ad esempio, è possibile effettuare `chroot` se si vuole testare le proprie dipendenze di compilazione. Oppure si può verificare come il pacchetto si comporta quando installato in un sistema di base puro. Gli strumenti di compilazione chroot usano questo pacchetto; si veda di seguito.

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` è un altro strumento per creazione di pacchetti automatizzato. Anch'esso può utilizzare ambienti chroot. Può essere utilizzato da solo, o come parte di un ambiente di compilazione distribuita in rete. In quest'ultimo caso, è parte del sistema utilizzato dagli autori di port per costruire pacchetti binari per tutte le architetture disponibili. Si consulti *wanna-build* per maggiori informazioni, e https://buildd.debian.org/ per vedere il sistema in azione.

9.5 Strumenti per caricare i pacchetti

I seguenti pacchetti consentono di automatizzare e semplificare il processo di caricamento dei pacchetti nell'archivio ufficiale.

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

Lo script `dcut` (parte del pacchetto `dput`, *dput*) aiuta a rimuovere i file dalla cartella ftp di caricamento.

9.6 Automazione della manutenzione

I seguenti strumenti aiutano ad automatizzare diverse attività di manutenzione, ad aggiungere le voci changelog o la firma e a guardare bug in Emacs facendo uso dei più nuovi e ufficiali `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` converte i pacchetti binari tra vari formati, tra cui Debian, RPM (RedHat), LSB (Linux Standard Base), Solaris e 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` (dal pacchetto `devscripts`, *devscripts*) esegue un comando sotto `strace` per determinare tutti i pacchetti che sono stati da esso utilizzati.

Per i pacchetti Debian, questo è utile quando si deve creare una voce `Build-Depends` per il proprio nuovo pacchetto: l'esecuzione del processo di generazione attraverso `dpkg-depcheck` fornirà una buona prima approssimazione delle dipendenze per la compilazione. Per esempio:

```
dpkg-depcheck -b debian/rules build
```

`dpkg-depcheck` può essere utilizzato anche per controllare le dipendenze in fase di esecuzione, soprattutto se il pacchetto usa exec 2 per eseguire altri programmi.

Per ulteriori informazioni consultare 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 Strumenti per i port

I seguenti strumenti sono utili per gli autori di port e per la cross-compilazione.

9.7.1 dpkg-cross

dpkg-cross è uno strumento per installare librerie e intestazioni per cross-compilare in modo simile a dpkg. Inoltre, le funzionalità di dpkg-buildpackage e dpkg-shlibdepssono state migliorate per supportare la cross-compilazione.

9.8 Documentazione ed informazioni

I seguenti pacchetti forniscono informazioni per i maintainer o aiutano con la scrittura di documentazione.

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).

Il pacchetto docbook-xsl fornisce i file XSL per l'applicazione di stili e la generazione dai sorgenti di vari output. Sara necessario un processore XSLT, come xsltproc, per utilizzare i fogli di stile XSL. La documentazione per i fogli di stile si può trovare nei vari pacchetti 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 *Mantenere la vostra chiave pubblica* 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.